

```
NNN      NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEEEEEEEEEEEEEE  TTTTTTTTTTTTTTTT  AAAAAAAAAA  CCCCCCCCCCCC  PPPPPPPPPPPP
NNN      NNN  EEE      TTT      AAA      AAA  CCC      PPP      PPP
NNN      NNN  EEE      TTT      AAA      AAA  CCC      FPP      PPP
NNN      NNN  EEE      TTT      AAA      AAA  CCC      PPP      PPP
NNNNNN    NNN  EEE      TTT      AAA      AAA  CCC      PPP      PPP
NNNNNN    NNN  EEE      TTT      AAA      AAA  CCC      PPP      PPP
NNNNNN    NNN  EEE      TTT      AAA      AAA  CCC      PPP      PPP
NNN      NNN  NNN  EEEEEEEEEEEEE  TTT      AAA      AAA  CCC      PPPPPPPPPPPP
NNN      NNN  NNN  EEEEEEEEEEEEE  TTT      AAA      AAA  CCC      PPPPPPPPPPPP
NNN      NNN  NNN  EEEEEEEEEEEEE  TTT      AAA      AAA  CCC      PPPPPPPPPPPP
NNN      NNNNNN  EEE      TTT      AAAAAAAAAAAAAAAAA  CCC      PPP
NNN      NNNNNN  EEE      TTT      AAAAAAAAAAAAAAAAA  CCC      PPP
NNN      NNNNNN  EEE      TTT      AAAAAAAAAAAAAAAAA  CCC      PPP
NNN      NNN      EEE      TTT      AAA      AAA  CCC      PPP
NNN      NNN      EEE      TTT      AAA      AAA  CCC      PPP
NNN      NNN      EEE      TTT      AAA      AAA  CCC      PPP
NNN      NNN      EEE      TTT      AAA      AAA  CCC      PPP
NNN      NNN      EEEEEEEEEEEEE  TTT      AAA      AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEE  TTT      AAA      AAA  CCCCCCCCCCCC  PPP
NNN      NNN      EEEEEEEEEEEEE  TTT      AAA      AAA  CCCCCCCCCCCC  PPP
```

NN	NN	EEEEEEEEEE	TTTTTTTTTT	CCCCCCCC	NN	NN	FFFFFFFFFF	AAAAAA	CCCCCCCC	TTTTTTTTTT	
NN	NN	EEEEEEEEEE	TTTTTTTTTT	CCCCCCCC	NN	NN	FFFFFFFFFF	AAAAAA	CCCCCCCC	TTTTTTTTTT	
NN	NN	EE	TT	CC	NN	NN	FF	AA	CC	TT	
NN	NN	EE	TT	CC	NN	NN	FF	AA	CC	TT	
NNNN	NN	EE	TT	CC	NNNN	NN	FF	AA	CC	TT	
NNNN	NN	EE	TT	CC	NNNN	NN	FF	AA	CC	TT	
NN	NN	EEEEEEEE	TT	CC	NN	NN	FFFFFFFF	AA	CC	TT	
NN	NN	EEEEEEEE	TT	CC	NN	NN	FFFFFFFF	AA	CC	TT	
NN	NNNN	EE	TT	CC	NN	NNNN	FF	AAAAAAAAAA	CC	TT	
NN	NNNN	EE	TT	CC	NN	NNNN	FF	AAAAAAAAAA	CC	TT	
NN	NN	EE	TT	CC	NN	NN	FF	AA	CC	TT	
NN	NN	EE	TT	CC	NN	NN	FF	AA	CC	TT	
NN	NN	EEEEEEEEEE	TT	CCCCCCCC	NN	NN	FF	AA	CCCCCCCC	TT	
NN	NN	EEEEEEEEEE	TT	CCCCCCCC	NN	NN	FF	AA	CCCCCCCC	TT	

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS

(2)	250	DECLARATIONS
(7)	440	NET\$SCAN_xxx - DEFAULT DATABASE SCANNER
(8)	512	NDIDEF_SCAN - DEFAULT NDI DATABASE SCANNER
(9)	681	NET\$SCAN_NDI - SCAN NDI DATABASE
(10)	992	NET\$SCAN_AJI - SCAN AJI DATABASE
(11)	1071	NET\$SCAN_SDI - SCAN SDI DATABASE
(12)	1196	NET\$SCAN_ARI - SCAN ARI DATABASE
(13)	1279	NET\$SPCSCAN_xxx - SPECIAL DATABASE SCAN ROUTINES
(14)	1314	NET\$SPCSCAN_NDI - SPECIAL SCAN OF NDI DATABASE
(15)	1419	NET\$PRE_QIO_xxx - PRE-QIO PROCESSING
(16)	1467	NET\$SHOW_xxx - PRE-SHOW PROCESSING
(17)	1499	NET\$DEFAULT_xxx - APPLY DEFAULT VALUES
(18)	1547	NET\$DEFAULT_NDI - APPLY DEFAULT VALUES TO NDI CNF
(19)	1578	NET\$INSERT_CNF - PRE-INSERTION PROCESSING
(20)	1748	NDI_MARKER - Insert executor NDI marker
(21)	1821	NET\$INSERT_NDI - PRE-INSERTION PROCESSING
(22)	2094	NET\$INSERT_OBI - PRE-INSERTION PROCESSING
(23)	2135	NET\$INSERT_xxx - PRE-INSERTION PROCESSING
(24)	2180	CHK_LOGIN_xxx - CHECK LOGIN STRING LENGTH
(25)	2226	NET\$SPCINS_xxx - SPECIAL DATABASE INSERTION ROUTINES
(25)	2227	NET\$SPCINS_DEF - DEFAULT DATABASE INSERTION ROUTINE
(26)	2278	NET\$SPCINS_NDI - INSERT NDI DATABASE INTO BINARY TREE
(27)	2315	NET\$DELETE_xxx - PRE-DELETE PROCESSING
(28)	2388	NET\$REMOVE_xxx - PROCESS THE REMOVE REQUEST
(28)	2389	NET\$REMOVE_DEF - DEFAULT PROCESSING OF THE REMOVE REQUEST
(29)	2480	SCAN_XWB - SCAN XWB LIST
(30)	2528	LNI_PARAMETER ACTION ROUTINES
(31)	2620	NDI_PARAMETER ACTION ROUTINES
(32)	3005	SUPPRESS_AREA - Suppress area from node address
(33)	3044	NET\$TEST_REACH - Test node reachability
(34)	3108	OBI_PARAMETER ACTION ROUTINES
(35)	3241	ESI_PARAMETER ACTION ROUTINES
(36)	3278	EFI_PARAMETER ACTION ROUTINES
(37)	3313	LLI_PARAMETER ACTION ROUTINES
(38)	3495	SPI_PARAMETER ACTION ROUTINES
(39)	3523	AJI_PARAMETER ACTION ROUTINES
(40)	3635	SDI_PARAMETER ACTION ROUTINES
(41)	3710	ARI_PARAMETER ACTION ROUTINES
(42)	3809	NET\$AREA_REACH - Test area reachability
(43)	3849	NET\$GET_EXEC_STA - GET EXECUTOR STATE
(44)	3873	NET\$NDI_BY_ADD - Find NDI CNF by node address
(45)	3921	NET\$LOCATE_NDI - Find phantom or real NDI CNF
(46)	3962	MOVE_PARAMETER SUBROUTINES
(47)	3978	FMT_CNT - FORMAT COUNTERS
(48)	4063	LOG_COUNTERS - LOG ZERO COUNTER EVENT

```
0000 1 .TITLE NETCNFACT - Configuration data base access action routines
0000 2 .IDENT 'V04-000'
0000 3 .DEFAULT DISPLACEMENT, LONG
0000 4
0000 5 *****
0000 6 *
0000 7 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 * ALL RIGHTS RESERVED.
0000 10 *
0000 11 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 * TRANSFERRED.
0000 17 *
0000 18 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 * CORPORATION.
0000 21 *
0000 22 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 *
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 : FACILITY: NETWORK ACP
0000 30
0000 31 : ABSTRACT:
0000 32 : This module provides support to the NETACP database management
0000 33 : including database entry insertion and action routines to
0000 34 : retrieve data for parameters which are not stored in any of
0000 35 : the CNF control blocks.
0000 36
0000 37 : ENVIRONMENT:
0000 38 : The module runs in kernel mode and at possibly elevated IPL.
0000 39 : It is therefore locked into the ACP's virtual address space
0000 40 : in order to prevent the need for paging.
0000 41
0000 42 : Since the ACP is work-queue driven, and since it is the ACP
0000 43 : that modifies the structure of the non-paged pool data base
0000 44 : including the RCB (actually a VCB) and the private structures
0000 45 : hanging off of the RCB, there is no need to obtain the system
0000 46 : data base mutex -- no races can occur. However, it is
0000 47 : necessary to raise IPL in order to stop any races with
0000 48 : NETDRIVER.
0000 49
0000 50
0000 51 : AUTHOR: A.Eldridge 14-Feb-80
0000 52
0000 53 : MODIFIED BY:
0000 54
0000 55 : V03-038 PRB0336 Paul Beck 24-Jun-1984 14:04
0000 56 : Allow SCSSYSTEMID match in area 1 without area in SCSSYSTEMID
0000 57 : SYSGEN parameter.
```

```
0000 58 :  
0000 59 :  
0000 60 : V03-037 RNG0037 Rod Gamache 18-Jun-1984  
0000 61 : Add logging of Data Base Entry re-used to log_counters routine.  
0000 62 : Fix termination of 'adjacency vector' search on NDI scan  
0000 63 : when max address is 1023.  
0000 64 :  
0000 65 : V03-036 PRB0326 Paul Beck 28-Mar-1984 15:51  
0000 66 : Use SYS$SYSTEM instead of SYS$SYSROOT:[SYSEXEC] for the default  
0000 67 : file spec in the FAB, to allow search lists to work correctly.  
0000 68 : Fix NDI_SCAN routine when used with LOOP NODES.  
0000 69 :  
0000 70 : V03-035 RNG0035 Rod Gamache 15-Mar-1984  
0000 71 : Fix routines that access new LLI structure to get the XWB  
0000 72 : address from the LLI first.  
0000 73 :  
0000 74 : V03-034 ADE0054 Alan D. Eldridge 16-Feb-1984  
0000 75 : Make changes to support converting the LLI to a 'real' database.  
0000 76 :  
0000 77 : V03-033 PRB0312 Paul Beck 4-Feb-1984 19:12  
0000 78 : Require local node name match SYSGEN parameters SCSNODEL/H  
0000 79 : if they are defined.  
0000 80 :  
0000 81 : V03-032 PRB0310 Paul Beck 26-Jan-1984 11:18  
0000 82 : Strip trailing ";" from parsed object filename string if it  
0000 83 : wasn't present in OBI,S,FID. This allows NETSERVER to make use  
0000 84 : of the installed version of an image (image activator will ignore  
0000 85 : installed version if explicit version number is specified).  
0000 86 : Also remove reference to NET$T_TSKFAB, no longer used.  
0000 87 :  
0000 88 : V031 RNG0031 Rod Gamache 13-Jan-1984  
0000 89 : Fix problem in previous fix, where it was attempting to  
0000 90 : delete the 'dummy NDI'.  
0000 91 :  
0000 92 : V030 RNG0030 Rod Gamache 14-Nov-1983  
0000 93 : Fix deletion of NDI data block when entry was re-inserted  
0000 94 : into binary trees.  
0000 95 :  
0000 96 : V029 TMH0029 Tim Halvorsen 10-Jul-1983  
0000 97 : Allow normal NDI entries to use the CIRCUIT parameter  
0000 98 : so that a user can explicitly specify the path to a  
0000 99 : node. This is different than loop nodes, which always  
0000 100 : use our own address so that they are looped back to us,  
0000 101 : but similar in that the CIRCUIT parameter is used.  
0000 102 : Add support for local alias addresses.  
0000 103 :  
0000 104 : V028 TMH0028 Tim Halvorsen 17-May-1983  
0000 105 : If we are an endnode, then return the designated router  
0000 106 : for the nearest level 2 router in the area database.  
0000 107 :  
0000 108 : V027 TMH0027 Tim Halvorsen 20-Apr-1983  
0000 109 : Add Service (DLE) database support.  
0000 110 : Don't return dummy NDI in special NDI scanner if the  
0000 111 : starting CNF is non-zero or not the CNR.  
0000 112 :  
0000 113 : V026 RNG0026 Rod Gamache 29-Mar-1983  
0000 114 : Add code to support the binary balanced trees for the  
NDI database.
```

```
0000 115 :  
0000 116 :  
0000 117 : V025 TMH0025 Tim Halvorsen 03-Mar-1983  
0000 118 : Use default filespec of SYS$SYSTEM for object procedure  
0000 119 : name if object name starts with a '$'.  
0000 120 : Handle new Level routing routing enable flag in RCB.  
0000 121 : Always return 'unreachable' if we cannot determine the  
0000 122 : 'next node to destination' of a remote node.  
0000 123 : For non-area routers, make area database consist of the  
0000 124 : local area, with cost/hops/nexthop referring to the 'nearest  
0000 125 : level 2 router'.  
0000 126 :  
0000 127 : V024 TMH0024 Tim Halvorsen 14-Feb-1983  
0000 128 : Remove node proxy access parameter.  
0000 129 : Add code so that the next hop on way to remote areas are  
0000 130 : returned with 'next hop to destination' and 'output circuit'.  
0000 131 : Add endnode key defaulting.  
0000 132 : Do not check new NDI address against executor max address  
0000 133 : if the NDI refers to another area, since our max address  
0000 134 : doesn't apply to other areas.  
0000 135 : Add support for EPIDs.  
0000 136 :  
0000 137 : V023 TMH0023 Tim Halvorsen 08-Jan-1983  
0000 138 : Fix some subroutine calls which were linked out of range.  
0000 139 :  
0000 140 : V022 TMH0022 Tim Halvorsen 17-Dec-1982  
0000 141 : Fix logical link scanning (for things like 'active  
0000 142 : links', etc.) so that it correctly matches the entire  
0000 143 : node address, including area number.  
0000 144 : Add LLI PNA action routine, which suppresses the area  
0000 145 : number in the node address if necessary.  
0000 146 : Reformat TAD search key as well as ADD search key so  
0000 147 : that addresses without areas matching the corresponding  
0000 148 : NDI object key.  
0000 149 : Optimize NDI_BY_ADD a bit.  
0000 150 :  
0000 151 : V021 TMH0021 Tim Halvorsen 05-Dec-1982  
0000 152 : Fix code which re-defines an NDI so that it correctly  
0000 153 : adjusts the NDI vector.  
0000 154 : Fix SHOW NODE nnn (by number).  
0000 155 : Add NNN parameter, which is the node name corresponding  
0000 156 : to NND (to speed up the SHOW KNOWN NODES request).  
0000 157 :  
0000 158 : V020 TMH0020 Tim Halvorsen 14-Oct-1982  
0000 159 : Add area routing support.  
0000 160 : If it cannot be determined if a node is reachable or not  
0000 161 : (because we are an endnode, or because its in another area),  
0000 162 : then return failure for the REA parameter ('don't know').  
0000 163 :  
0000 164 : V019 TMH0019 Tim Halvorsen 01-Oct-1982  
0000 165 : Make Phase II nodes 1 hop away, even though the minimum  
0000 166 : cost/hops vector says it's unreachable (it's unreachable  
0000 167 : in the vector so that the routing messages to other nodes  
0000 168 : reflect it).  
0000 169 :  
0000 170 : V018 TMH0018 Tim Halvorsen 16-Sep-1982  
0000 171 : Change name of routine to reset automatic counter timers.  
Add support for AJI Adjacency database.
```

0000	172	:			
0000	173	:	V017	TMH0017	Tim Halvorsen 01-Jul-1982
0000	174	:			Add node action routine NND, to return the node address
0000	175	:			of the next node in the path to the remote node.
0000	176	:			Add executor PHA action routine, to return the NI physical
0000	177	:			address used by the current node.
0000	178	:			Modify TEST_REACH to return the ADJ index, rather than
0000	179	:			the LPD index, and modify all callers to lookup the
0000	180	:			appropriate information in the ADJ block.
0000	181	:			
0000	182	:	V016	TMH0016	Tim Halvorsen 30-Jun-1982
0000	183	:			Add entry point for applying a set of default values
0000	184	:			given a default table address.
0000	185	:			
0000	186	:	V015	TMH0015	Tim Halvorsen 16-Jun-1982
0000	187	:			Add support for SPI database.
0000	188	:			Fix a second bug in the logical link collating sequence,
0000	189	:			which caused a loop in the database traversal.
0000	190	:			
0000	191	:	V014	TMH0014	Tim Halvorsen 04-Apr-1982
0000	192	:			Remove all code specific to CRI and PLI databases, and
0000	193	:			move it into a new module NETCNFDLL.
0000	194	:			Remove all explicit displacement specifiers from operands,
0000	195	:			and make the default = word for the entire module.
0000	196	:			Change all CNF action routines to use the new action routine
0000	197	:			interface (NETCNF now automatically allocates a TMP buffer).
0000	198	:			Remove obsolete NUL action routines.
0000	199	:			Rename CNFST_MASK to CNFSL_MASK.
0000	200	:			Fix bug in logical link collating order, which sometimes
0000	201	:			caused a loop (sometimes finite) in the SHOW KNOWN LINKS
0000	202	:			display.
0000	203	:			Rewrite NETSTEST_REACH to use NETSFIND_LPD to get LPD address.
0000	204	:			
0000	205	:	V013	TMH0013	Tim Halvorsen 27-Mar-1982
0000	206	:			Fix code to translate an NMA parameter code returned by
0000	207	:			a datalink driver validation error to a NFB code.
0000	208	:			
0000	209	:	V02-12	ADE0052	A.Eldridge 25-Jan-82
0000	210	:			Get the number of DMC receive buffers from the PLI database
0000	211	:			instead of the CRI database.
0000	212	:			
0000	213	:	V02-11	ADE0051	A.Eldridge 22-Jan-82
0000	214	:			Disallow NFBSC_NDI_PRX values of "both" or "outbound" if
0000	215	:			explicit outbound defaults exist.
0000	216	:			
0000	217	:	V02-10	ADE0050	A.Eldridge 19-Jan-82
0000	218	:			Added routine NET\$APPLY_DFLT which applies default values
0000	219	:			to selected CNF parameters.
0000	220	:			
0000	221	:	V02-09	ADE0044	A.Eldridge 06-Jan-82
0000	222	:			Removed the 'retransmit timer' (RTT) parameter from the
0000	223	:			circuit database
0000	224	:			
0000	225	:	V02-08	ADE0043	A.Eldridge 31-Dec-81
0000	226	:			Rename the CI DECnet class driver mnemonic to "CN".
0000	227	:			
0000	228	:	V02-07	ADE0042	A.Eldridge 22-Dec-81

0000	229	:					Added support for the logical-link 'RID' field
0000	230	:					
0000	231	:	V02-06	ADE0041	A.Eldridge	14-Dec-81	
0000	232	:					Added support for line counters.
0000	233	:					
0000	234	:	V02-05	ADE0040	A.Eldridge	11-Dec-81	
0000	235	:					Fix node counter bug that was returning the Executor node
0000	236	:					counters for nodes which are unreachable.
0000	237	:					
0000	238	:	V02-04	ADE0030	A.Eldridge	30-Nov-81	
0000	239	:					Added support for zero counter event.
0000	240	:					
0000	241	:	V02-03	ADE0029	A.Eldridge	21-Jul-81	
0000	242	:					Replace datalink (DLI) database with circuit (CRI) and
0000	243	:					physical line (PLI) databases.
0000	244	:					
0000	245	:	V02-02	ADE0028	A.Eldridge	21-Jul-81	
0000	246	:					Updated to support modified CNF data base interface.
0000	247	:					
0000	248	:					


```
0000 250      .SBTTL  DECLARATIONS
0000 251      ;
0000 252      ; INCLUDE FILES
0000 253      ;
0000 254      $FABDEF      ; File Access Block
0000 255      $NAMDEF      ; File name block
0000 256      $JPIDEF      ; $GETJPI definitions
0000 257
0000 258      $NFBDEF      ; Network Function Block (ACP control QIO definitions)
0000 259      $NMADEF      ; Network Management (NICE protocol) definitions
0000 260      $EVCDEF      ; DECnet Event logging symbols
0000 261      $DRDEF      ; Disconnect reason codes
0000 262      $CNRDEF      ; Configuration Root block
0000 263      $CNFDEF      ; Configuration data block
0000 264      $ICBDEF      ; Internal Connect Block
0000 265      $LLIDEF      ; Logical link information block
0000 266      $LNIDEF      ; Local node information
0000 267      $LPDDEF      ; Logical path descriptor
0000 268      $ADJDEF      ; Adjacency control block
0000 269      $LTBDEF      ; Logical-link table
0000 270      $NETSYMDEF    ; Miscellaneous network symbols
0000 271      $NETUPDDEF    ; Symbols used in private NETACP interface to NETDRIVER
0000 272      $NSPMSGDEF    ; DNA architecture definitions & message formats
0000 273      $NDIDEF      ; Node information block
0000 274      $RCBDEF      ; Routing Control Block (analogous to Volume Control
0000 275      ; block)
0000 276      $WQEDEF      ; Work Queue Element
0000 277      $XWBDEF      ; Network Window Block -- logical-link context block
0000 278      $DWBDEF      ; DLE window block
0000 279      $UCBDEF      ; UCB definitions
0000 280
00000090 0000 281      UCBSQ_DWB_LIST = UCB$C_LENGTH      ; && TEMP - MUST BE SAME AS NDDRIVER
0000 282
0000 283      ;
0000 284      ; EQUATED SYMBOLS:
0000 285      ;
0000 286
00000024 0000 287      NDI_ADD = CNF$C_LENGTH+NDISW_ADD      ; Define symbol for convenience
0000 288
0000 289      ;
0000 290      ; Define special CNF flags for each database
0000 291      ;
00000004 0000 292      NDI_V_LOOP      = CNF$V_FLG_MRK1      ; Set for "loop" nodes
00000005 0000 293      NDI_V_LOCAL      = CNF$V_FLG_MRK2      ; Set for the "local" node
00000006 0000 294      NDI_V_MARKER      = CNF$V_FLG_MRK3      ; Set for "marker" node
```

```
0000 296 :  
0000 297 : OWN STORAGE  
0000 298 :  
0000 299 :  
0000 300 .PSECT NET_IMPURE,WRT,NOEXE, LONG  
0000 301  
00000000 0000 302 NDI_L_NACS: .LONG 0 ; Remember number of non-null access  
00000016 0004 303 NDI_Z_COL: .BLKB 15+3 ; Size of collate string, Device name  
0016 304 ; + node id + size.  
0016 305 .ALIGN LONG  
00000000 00000000 0018 306 NDI_Q_NAME: .QUAD 0 ; Descriptor of new nodename  
00000000 00000000 0020 307 NDI_Q_LNAME: .QUAD 0 ; Descriptor of new logical nodename  
00000038 0028 308 NDI_LNAMEBUF: .BLKB 16 ; Buffer for build logical node-name  
00000040 0038 309 IOSB: .BLKB 8 ; I/O status block  
0040 310  
0040 311  
0000 312 .PSECT NET_PURE,NOWRT,NOEXE, LONG  
0000 313  
0000 314  
4F 4E 24 53 59 53 00000008'010E0000' 0000 315 SYSNODE_DESC: .ASCID 'SYS$NODE' ; Descriptor for logical name  
45 44 000E  
4C 43 24 53 59 53 00000018'010E0000' 0010 316 CLUNODE_DESC: .ASCID 'SYS$CLUSTER_NODE' ; Descriptor for logical name  
45 44 4F 4E 5F 52 45 54 53 55 001E  
0028 317  
0028 318 NDI_NLOGIN_VEC: ; Vector of NDI nonpriv login field id's  
0028 319 .CNFFLD ndi,s,nus ; nonpriv user  
002C 320 .CNFFLD ndi,s,npw ; nonpriv password  
0030 321 .CNFFLD ndi,s,nac ; nonpriv account  
00000000 0034 322 .LONG 0 ; Terminate the vecvtor  
0038 323  
0038 324 NDI_PLOGIN_VEC: ; Vector of NDI priv login field id's  
0038 325 .CNFFLD ndi,s,pus ; priv user  
003C 326 .CNFFLD ndi,s,ppw ; priv password  
0040 327 .CNFFLD ndi,s,pac ; priv account  
00000000 0044 328 .LONG 0 ; Terminate the vecvtor  
0048 329  
0048 330 OBI_LOGIN_VEC: ; Vector of OBI login field id's  
0048 331 .CNFFLD obi,s,usr ; user  
004C 332 .CNFFLD obi,s,psw ; password  
0050 333 .CNFFLD obi,s,acc ; account  
00000000 0054 334 .LONG 0 ; Terminate the vecvtor  
0058 335
```

```
0058 337
0058 338
0058 339 : The following macros build a conversion table for formatting counters
0058 340 : into NICE format. Each counter i.d. contain is bit encoded to contain
0058 341 : formatting information as follows:
0058 342 :
0058 343 :   15  14  13  12  11      0      Bit
0058 344 :   < 1 >< width >< 0 >< counter i.d. >      Field
0058 345 :
0058 346 :
0058 347 :
00000001 0058 348 $WIDTH_B = 1 ; Counter width specifier for bytes
00000002 0058 349 $WIDTH_W = 2 ; Counter width specifier for words
00000003 0058 350 $WIDTH_L = 3 ; Counter width specifier for longwords
0058 351
00000000 0058 352 NETSC_NMACNT_SLZ = <1@15>!<<$WIDTH_W>@13>!0 ; Seconds since last zeroed
0058 353
0058 354 .MACRO $COUNT_ENT base,nice,pre,mod,count,width; Insert table entry
0058 355 :
0058 356 .WORD <1@15>!<<$WIDTH_'width'>@13>!-- Counter flag, Counter width
0058 357 <NMAC_'nice'_'count'> Nice counter i.d.
0058 358 .WORD 'pre'$'width'_'mod'_'count' - Offset into internal structure
0058 359 - base minus internal structure base
0058 360 .ENDM $COUNT_ENT
0058 361
0058 362 .MACRO $COUNT_TAB base,nice,pre,mod,list ; Create counter formatting table
0058 363 :
0058 364 .IRP A,<list>
0058 365 $COUNT_ENT base,nice,pre,mod,A ; Insert table entry
0058 366 .ENDR
0058 367 :
0058 368 .LONG 0 ; Terminate the table
0058 369
0058 370 .ENDM $COUNT_TAB
0058 371
0058 372
0058 373
0058 374
00000064 0058 375 CNT_FMT_BUFSIZ = 100 ; Size required to accomodate largest formatted
0058 376 ; counter buffer
0058 377
00000014 0058 378 NDC$ _MRC = NDC$ _PRC ;& Setup synonyms until NETNPAGED.MDL is fixed
00000018 0058 379 NDC$ _MSN = NDC$ _PSN
0058 380
0058 381 NDC_CNT_TAB: ; Common node counter table
0058 382
0058 383 $COUNT_TAB NDC$ _ABS _TIM,CTNOD,NDC,,-
0058 384 <-
0058 385 <BRC,L>,- ; Bytes received
0058 386 <BSN,L>,- ; Bytes sent
0058 387 <MRC,L>,- ; Packets received
0058 388 <MSN,L>,- ; Packets sent
0058 389 <CRC,W>,- ; Connects received
0058 390 <CSN,W>,- ; Connects sent
0058 391 <RTO,W>,- ; Response timeouts
0058 392 <RSE,W>,- ; Transmitted connect rejects due to resource
0058 393 - ; errors
```

```

0058 394 >
007C 395
007C 396 RCB_CNT_TAB: ; Local node counters
007C 397
007C 398 $COUNT_TAB RCB$$_ABS_TIM,CTNOD,RCB,CNT_,-
007C 399 <-
007C 400 <MLL,W>,- ; Maximum logical links active
007C 401 <APL,B>,- ; Aged packet loss
007C 402 <NUL,W>,- ; Node unreachable packet loss
007C 403 <NOL,B>,- ; Node out-of-range packet loss
007C 404 <OPL,B>,- ; Oversized packet loss
007C 405 <PFE,B>,- ; Packet format error
007C 406 <RUL,B>,- ; Partial routing update loss
007C 407 <VER,B>,- ; Verification rejects
007C 408 <XRE,W>,- ; Xmitted connect resource errors
007C 409 > -;

```

```
00000040 411 .PSECT NET_IMPURE,WRT,NOEXE
0040 412
0040 413
00000000 0040 414 UNAMES: .LONG 0 ; Returns resultant user name length
00000050 0044 415 UNAME: .BLKB 12 ; Returns user name
00000000 0050 416 PNAMES: .LONG 0 ; Returns resultant process name length
00000064 0054 417 PNAME: .BLKB 16 ; Returns process name
0064 418 .ALIGN LONG
0064 419
000C 0064 420 ITEM_LIST: ; $GETJPI item list for logical links
0202 0066 421 .WORD 12 ; Size of username buffer
00000044' 0068 422 .WORD JPIS_USERNAME ; I.d. of username parameter
00000040' 006C 423 .ADDRESS UNAME ; Address of username buffer
0070 424 .ADDRESS UNAMES ; Address of buffer to return length
000F 0070 425
031C 0072 426 .WORD 15 ; Size of process name buffer
00000054' 0074 427 .WORD JPIS_PRCNAM ; I.d. of process name parameter
00000050' 0078 428 .ADDRESS PNAME ; Address of process name buffer
007C 429 .ADDRESS PNAMES ; Address of buffer to return length
00000000 007C 430
0080 431 .LONG 0 ; Terminate the list
0080 432
0080 433
0080 434 NETST_PRSNAM: $NAM ESS = 255
00E0 435 NETST_SYSFAB: $FAB DNM = <SYSS$SYSTEM:.COM>,-
00E0 436 NAM = NETST_PRSNAM
0130 437
00000000 438 .PSECT NET_CODE,NOWRT,EXE
```

```
0000 440      .SBTTL NET$SCAN_XXX - DEFAULT DATABASE SCANNER
0000 441      :+
0000 442      : NET$SCAN_XXX - Scan database
0000 443      :
0000 444      : This co-routine is used to scan the database, and return to the caller
0000 445      : (co-routine) for each entry in the database. These routines establish
0000 446      : the order of the database entries, above that of the natural ordering of
0000 447      : the collating field.
0000 448      :
0000 449      : Inputs:
0000 450      :
0000 451      :     R11 = Address of CNR
0000 452      :     R10 = Address of starting CNF (or 0 if to start at the beginning)
0000 453      :
0000 454      : Outputs:
0000 455      :
0000 456      :     R10 = Address of CNF if dialogue aborted prematurely, else 0.
0000 457      :
0000 458      : The caller receives control on each database entry in list (via co-routine
0000 459      : call).
0000 460      :
0000 461      : On input to co-routine:
0000 462      :
0000 463      :     R0 = True if entry was found. False if at end of list (R10 invalid)
0000 464      :     R10 = Address of CNF entry found
0000 465      :
0000 466      : On output from co-routine:
0000 467      :
0000 468      :     R0 = CNFS_ADVANCE      Advance to next CNF, continue dialogue
0000 469      :     CNFS_TAKE_PREV    Return previous CNF, abort dialogue
0000 470      :     CNFS_TAKE_CURR    Return current CNF, abort dialogue
0000 471      :     CNFS_QUIT          Return no CNF (R10 = 0), abort dialogue
0000 472      :
0000 473      : *** These routines must be abortable via a RET ***
0000 474      : ---
0000 475      :
0000 476      NET$SCAN_LLI::      ; Logical-link scanner co-routine
0000 477      NET$SCAN_LNI::      ; Local node CNF scanner co-routine
0000 478      NET$SCAN_OBI::      ; Object CNF scanner co-routine
0000 479      NET$SCAN_EFI::      ; Event filter CNF scanner co-routine
0000 480      NET$SCAN_ESI::      ; Event sink CNF scanner co-routine
0000 481      NET$SCAN_SPI::      ; Server process CNF scanner co-routine
0000 482      DEFAULT_SCAN::     ; Default CNF scanner co-routine
0000 483      :
0000 484      ASSUME CNF$SL_FLINK EQ 0
0000 485      ASSUME CNF$SL_FLINK EQ CNR$SL_FLINK
0000 486      ASSUME CNF$SB_FLG  EQ CNR$SB_FLG
0000 487      :
0000 488      TSTL R10      ; Already pointing to a CNF ?
0000 489      BNEQ 10$      ; If NEQ then yes
0000 490      MOVAB CNR$SL_FLINK(R11),R10 ; Get address of ptr to 1st CNF
0000 491      MOVL #1,R0    ; Indicate success
0000 492      JSB @($P)+    ; Call back our caller
0000 493      $DISPATCH R0,<-
0000 494      :
0000 495      <CNFS_ADVANCE, 30$>    -; Advance to next CNF, continue dialogue
0000 496      <CNFS_TAKE_PREV, 40$> -; Return previous CNF, abort dialogue
```

5A 50 03 6B 01 9E 05 12 9E 16

			000C	497		<CNFS_QUIT,	50\$>	-;	CNF not found, abort dialogue
			000C	498		<CNFS_TAKE_CURR,	60\$>	-;	Take current CNF, abort dialogue
			000C	499		>			
			0018	500		BUG_CHECK	NETNOSTATE,FATAL		
			001C	501					
SA	6A	D0	001C	502	30\$:	MOVL	CNFS_L_FLINK(R10),R10	:	Advance to next CNF
	00	E1	001F	503		BBC	#CNFS_V_FLG_CNR,-	:	
E3	0B	AA	0021	504			CNFS_B_FLG(R10),10\$	:	If BC, then R10 is not the CNR
	50	D4	0024	505		CLRL	R0	:	Say 'no more CNFs'
	E2	11	0026	506		BRB	20\$	:	Call back with the bad news
SA	04	AA	D0	0028	40\$:	MOVL	CNFS_L_BLINK(R10),R10	:	Go back to previous CNF
	02	11	002C	508		BRB	60\$	:	Continue
	5A	D4	002E	509	50\$:	CLRL	R10	:	Nullify CNF pointer
		05	0030	510	60\$:	RSB		:	Return to caller, terminate dialogue

```
0031 512 .SBTTL NDIDEF_SCAN - DEFAULT NDI DATABASE SCANNER
0031 513 :+
0031 514 : NDIDEF_SCAN - Default NDI database scanner
0031 515 :
0031 516 : This co-routine is used to scan the database, and return to the caller
0031 517 : (co-routine) for each entry in the database. THIS routine establishes
0031 518 : the order of the database entries, above that of the natural ordering of
0031 519 : the collating field.
0031 520 :
0031 521 : Inputs:
0031 522 :
0031 523 :     R11 = Address of CNR
0031 524 :     R10 = Address of starting CNF (or 0 if to start at the beginning)
0031 525 :
0031 526 : Outputs:
0031 527 :
0031 528 :     R10 = Address of CNF if dialogue aborted prematurely, else 0.
0031 529 :
0031 530 : The caller receives control on each database entry in list (via co-routine
0031 531 : call).
0031 532 :
0031 533 : On input to co-routine:
0031 534 :
0031 535 :     R0 = True if entry was found. False if at end of list (R10 invalid)
0031 536 :     R10 = Address of CNF entry found
0031 537 :
0031 538 : On output from co-routine:
0031 539 :
0031 540 :     R0 = CNFS_ADVANCE      Advance to next CNF, continue dialogue
0031 541 :         CNFS_TAKE_PREV    Return previous CNF, abort dialogue
0031 542 :         CNFS_TAKE_CURR    Return current CNF, abort dialogue
0031 543 :         CNFS_QUIT         Return no CNF (R10 = 0), abort dialogue
0031 544 :
0031 545 :     R1,R2 are destroyed.
0031 546 :
0031 547 : *** These routines must be abortable via a RET ***
0031 548 : ---
0031 549 :
0031 550 NDIDEF_SCAN:: : Default NDI scanner co-routine
0031 551 :
0031 552 :     Allocate some storage on the stack to hold the last NDI
0031 553 :     returned to the caller. This makes backing up to the
0031 554 :     previous entry very easy.
0031 555 :
00000000 0031 556 :     ORIGAP = 00
00000004 0031 557 :     CALLER = 04
00000008 0031 558 :     BTECOR = 08
0000000C 0031 559 :     TEMPRG = 12
00000014 0031 560 :     CNFADD = 20
00000018 0031 561 :     NODADD = 24
0031 562 :     CLRQ   -(SP) : CNFADD(AP) = CNF address.
0031 563 :     :          : NODADD(AP) = Node number (in vector)
0031 564 :     CLRQ   -(SP) : Make room on stack for temp save regs
0031 565 :     :          : REF: TEMPRG(AP)
00000000'EF 9F 0031 566 :     PUSHAB NET$TRAVERSE_NDI : Push next co-routine address
0031 567 :     :          : REF: BTECOR(AP)
0031 568 :     CLRL   -(SP) : Return address to caller
```



```

      SC DD 003D 569
      SC SE DO 003D 570
      1C AE DD 003F 571
      DD 0042 572
      DD 0045 573
      DD 0045 574
      DD 0045 575
      DD 0045 576
      DD 0045 577
      5A D5 0045 578
      03 12 0047 579
      5A 5B DO 0049 580
      DD 004C 581
      DD 004C 582
      DD 004C 583
      50 01 DO 004C 584
      9E 16 004F 585
      DD 0051 586
      DD 0051 587
      DD 0051 588
      DD 0051 589
      DD 0051 590
      5B 5A D1 0051 591
      51 13 0054 592
      04 AC 8E DO 0056 593
      14 AC 5A DO 005A 594
      18 AC 12 AA BO 005E 595
      08 AC 00000000'EF 9E 0063 596
      0C AC 57 7D 006B 597
      DD 006F 598
      DD 006F 599
      DD 006F 600
      58 00000004'EF 9E 006F 601
      53 5B DO 0076 602
      DD 0079 603
      0367 8F BB 0079 604
      0E44 30 007D 605
      0367 8F BA 0080 606
      DD 0084 607
      57 53 5B C3 0084 608
      DD 0088 609
      DD 0088 610
      00D7'CF 9F 0088 611
      00000000'EF 16 008C 612
      57 0C AC 7D 0092 613
      29 50 E9 0096 614
      22 50 01 E0 0099 615
      51 000000D7'EF 9E 009D 616
      002D 31 00A4 617
      DD 00A7 618
      DD 00A7 619
      DD 00A7 620
      DD 00A7 621
      DD 00A7 622
      DD 00A7 623
      DD 00A7 624
      DD 00A7 625

      PUSHL AP ; REF: CALLER(AP)
      MOVL SP,AP ; Save the AP
      PUSHL 28(SP) ; Save current stack pointer
      ; Copy return address to caller

      ; Initialize "last CNF" pointer. This is the pointer to the last
      ; CNF processed and may actually be the CNR.

      TSTL R10 ; Is there a current NDI
      BNEQ 5$ ; If NEQ then yes
      MOVL R11,R10 ; Else start at the head of the list

      ; Return to caller with "initialization complete"

      MOVL #1,R0 ; Initialization successful
      JSB @($P)+ ; Call back the caller, on return
      ; R0 = function to perform

      ; The following code insures that if we are not called recursively,
      ; then we will always find the next NDI after the last one.

      CMPL R10,R11 ; Are we starting from beginning?
      BEQL 8$ ; Br if yes, okay to proceed
      MOVL ($P)+,CALLER(AP) ; Save caller's return address
      MOVL R10,CNFADD(AP) ; Save last CNF returned
      MOVW CNF$W ID(R10),NODADD(AP) ; Save last node address returned
      MOVAB NET$TRAVERSE_ALT,BTECOR(AP) ; Set address of resume code
      MOVQ R7,TEMPRG(AP) ; Save R7, R8

      ; Get the collating value for this NDI (R10)

      MOVAB NDI_Z_COL,R8 ; Get address of collate buffer
      MOVL R8,R3 ; Copy output buffer address

      PUSHR #^M<R0,R1,R2,R5,R6,R8,R9> ; Save registers
      BSBW NET$NDI_S_COL ; Get the collating value
      POPR #^M<R0,R1,R2,R5,R6,R8,R9> ; Restore registers

      SUBL3 R8,R3,R7 ; Calculate length of collate string

      PUSHAB W^160$ ; Push address of return
      JSB NET$RESUME_NDI ; Call routine to build up stack
      MOVQ TEMPRG(AP),R7 ; Restore R7, R8
      BLBC R0,20$ ; Br if failure to proceed
      BBS #1,R0,10$ ; Br if take current
      MOVAB 160$,R1 ; Store return address
      BRW 140$ ; And continue processing

      ; Engage in a co-routine dialogue with the user. The scanner half
      ; of the dialogue owns the stack until the calling routine calls
      ; back with any function code other than CNF$ADVANCE; the other
      ; function codes causes the scanner to return to the caller with a
      ; clean stack thus terminating the co-routine dialogue.

      ; Register usage:
      ; R1 may be destroyed, by this routine, but must be preserved
```

```
00A7 626 : on calls to the co-routine calls to the SCAN routine. Therefore
00A7 627 : on initial input R1 will be zero, but will be what the caller
00A7 628 : requires on output. The same goes for R2.
00A7 629 :
04 AC 8E DO 00A7 630 8$: MOVL (SP)+,CALLER(AP) ; Save callers address on stack
51 OC AC 7D 00AB 631 MOVQ TEMPRG(AP),R1 ; Restore R1,R2
00AF 632 $DISPATCH RO,- ; Dispatch on function code returned by
00AF 633 <- ; co-routine
00AF 634 <CNFS_ADVANCE, 100$>,- ; Advance to next CNF, continue dialogue
00AF 635 <CNFS_TAKE_PREV, 200$>,- ; Return previous CNF, abort dialogue
00AF 636 <CNFS_QUIT, 300$>,- ; CNF not found, abort dialogue
00AF 637 <CNFS_TAKE_CURR, 400$>,- ; Take current CNF, abort dialogue
00AF 638 >
00BB 639 BUG_CHECK NETNOSTATE,FATAL
00BF 640
50 01 DO 00BF 641 10$: MOVL #1,R0 ; Indicate success
OC AC 51 7D 00C2 642
04 BC 16 00C2 643 20$: MOVQ R1,TEMPRG(AP) ; Save R1,R2
DC 11 00C6 644 JSB @CALLER(AP) ; Call back the caller with status
00CB 645 BRB 8$
00CB 646
00CB 647 100$: :
00CB 648 : Advance to the next CNF.
00CB 649 :
14 AC 5A DO 00CB 650 MOVL R10,CNFADD(AP) ; Save last CNF given back to caller
18 AC 12 AA BO 00CF 651 MOVW CNF$W_ID(R10),NODADD(AP); Save last node # given to caller
00D4 652 :
00D4 653 : Skip to next node in data base
00D4 654 :
08 BC 16 00D4 655 140$: JSB @BTECOR(AP) ; Skip to next node, call BTE co-routine
E8 50 E9 00D7 656 160$: BLBC RO,20$ ; Br if error, don't pop stack
08 AC 8E DO 00DA 657 MOVL (SP)+,BTECOR(AP) ; Else, save return address
06 E0 00DE 658 BBS #NDI_V_MARKER,- ; Never return the marker CNF
F1 OB AA 00E0 659 CNF$B_FLG(R10),140$
DA 11 00E3 660 BRB 10$ ; And return CNF
00E5 661 :
00E5 662 200$: :
00E5 663 : The caller wants to take the previous CNF
00E5 664 :
5A 14 AC DO 00E5 665 MOVL CNFADD(AP),R10 ; Get previous CNF address
02 12 00E9 666 BNEQ 400$ ; Branch if none
00EB 667 :
00EB 668 : The caller wants to call it quits
00EB 669 :
5A D4 00EB 670 300$: CLRL R10 ; Nullify CNF pointer
00ED 671 :
00ED 672 : The caller is done with the scan and wants the stack back.
00ED 673 :
50 04 AC DO 00ED 674 400$: MOVL CALLER(AP),R0 ; Get caller's return address
5E 5C DO 00F1 675 MOVQ AP,SP ; Restore original stack pointer
5C 8ED0 00F4 676 POPL AP ; Restore original AP
5E 1C CO 00F7 677 ADDL #7*4,SP ; Pop scratch storage
60 17 00FA 678 JMP (R0) ; Return to caller
00FC 679
```

```
00FC 681 .SBTTL NET$SCAN_NDI - SCAN NDI DATABASE
00FC 682 :+
00FC 683 : NET$SCAN_NDI - Scan NDI database
00FC 684 :
00FC 685 : This co-routine is used to scan the database, and return to the caller
00FC 686 : (co-routine) for each entry in the database. These routines establish
00FC 687 : the order of the database entries, above that of the natural ordering of
00FC 688 : the collating field.
00FC 689 :
00FC 690 : Inputs:
00FC 691 :
00FC 692 :     R11 = Address of CNR
00FC 693 :     R10 = Address of starting CNF (or 0 if to start at the beginning)
00FC 694 :
00FC 695 : Outputs:
00FC 696 :
00FC 697 :     R10 = Address of CNF if dialogue aborted prematurely, else 0.
00FC 698 :
00FC 699 : The caller receives control on each database entry in list (via co-routine
00FC 700 : call).
00FC 701 :
00FC 702 : On input to co-routine:
00FC 703 :
00FC 704 :     R0 = True if entry was found. False if at end of list (R10 invalid)
00FC 705 :     R10 = Address of CNF entry found
00FC 706 :
00FC 707 : On output from co-routine:
00FC 708 :
00FC 709 :     R0 = CNFS_ADVANCE      Advance to next CNF, continue dialogue
00FC 710 :         CNFS_TAKE_PREV    Return previous CNF, abort dialogue
00FC 711 :         CNFS_TAKE_CURR    Return current CNF, abort dialogue
00FC 712 :         CNFS_QUIT         Return no CNF (R10 = 0), abort dialogue
00FC 713 :
00FC 714 :     R1,R2 are destroyed.
00FC 715 :
00FC 716 : *** These routines must be abortable via a RET ***
00FC 717 : ---
00FC 718 :
00FC 719 NET$SCAN_NDI::                                : Find next NDI block
00FC 720 :
00FC 721 ASSUME CNF$SL_FLINK EQ 0
00FC 722 ASSUME CNF$SL_FLINK EQ CNR$SL_FLINK
00FC 723 ASSUME CNF$SB_FLG EQ CNR$SB_FLG
00FC 724 :
00FC 725 BBC #NET$V_INTRNL - : If BC then external and so we are
00FE 726 NET$GL_FLAGS,10$ : interested in "phantom" NDI CNFs
0104 727 BRW NDI$DEF_SCAN : Else, only "real" NDI CNFs
0107 728 10$:
0107 729 :
0107 730 : Allocate some storage on the stack to hold the last NDI
0107 731 : returned to the caller. This makes backing up to the
0107 732 : previous entry very easy.
0107 733 :
0107 734 ORIGAP = 00
0107 735 CALLER = 04
0107 736 BTECOR = 08
0107 737 SAVREG = 12
0107 738 TEMPRG = 20
```

09 E1
03 00000000'EF
FF2A 31

00000000
00000004
00000008
0000000C
00000014

```
0000001C 0107 738 CNFADD = 28
00000020 0107 739 NODADD = 32
7E 7C 0107 740 CLRQ -(SP) ; CNFADD(AP) = CNF address.
0109 741 ; NODADD(AP) = Node number (in vector)
7E 7C 0109 742 CLRQ -(SP) ; Make room on stack for temp save regs
7E 7C 010B 743 CLRQ -(SP) ; REF: TEMPRG(AP)
00000000'EF 9F 010D 744 PUSHAB NET$TRAVERSE_NDI ; Push next co-routine address
0113 745 ; REF: BTECOR(AP)
7E D4 0113 746 CLRL -(SP) ; Caller's return address
0115 747 ; REF: CALLER(AP)
5C 5C DD 0115 748 PUSHL AP ; Save the AP
5C 5E DD 0117 749 MOVL SP,AP ; Save current stack pointer
24 AE DD 011A 750 PUSHL 36(SP) ; Copy return address to caller
011D 751 ;
011D 752 ;
011D 753 ; Initialize "last CNF" pointer. This is the pointer to the last
011D 754 ; CNF processed and may a lly be the CNR.
011D 755 ;
5A D5 011D 756 TSTL R10 ; Is there a current NDI
03 12 011F 757 BNEQ 20$ ; If NEQ then yes
5A 5B DD 0121 758 MOVL R11,R10 ; Else start at the head of the list
0124 759 20$: ;
0124 760 ; Return to caller with "initialization complete"
0124 761 ;
50 01 DD 0124 762 MOVL #1,R0 ; Initialization successful
9E 16 0127 763 JSB @($P)+ ; Call back the caller, on return
0129 764 ; R0 = function to perform
0129 765 ;
0129 766 ; The following code insures that if we are not called recursively,
0129 767 ; then we will always find the next NDI after the last one.
0129 768 ;
5B 5A D1 0129 769 CMPL R10,R11 ; Are we starting from beginning?
55 13 012C 770 BEQL 40$ ; Br if yes, okay to proceed
50 00 D1 012E 771 CMPL #CNF$_ADVANCE,R0 ; Are we advancing?
50 50 12 0131 772 BNEQ 40$ ; Br if not, don't have to setup stack
04 AC 8E DD 0133 773 MOVL (SP)+,CALLER(AP) ; Save caller's return address
14 AC 53 7D 0137 774 MOVQ R3,TEMPRG(AP) ; Save R3,R4
08 AC 00000000'EF 9E 013B 775 MOVAB NET$TRAVERSE_ALT,BTECOR(AP) ; Set address of resume code
0C AC 57 7D 0143 776 MOVQ R7,SAVREG(AP) ; Save R7, R8
0147 777 ;
0147 778 ; Get the collating value for this NDI (R10)
0147 779 ;
0156 30 0147 780 BSBW GET_COLLATE ; Get the collate value from NDI
014A 781 ;
014A 782 ; Contine where we left off in previous call, use collate value to
014A 783 ; find BTE entries to rebuild the stack for subsequent calls.
014A 784 ;
54 5A DD 014A 785 MOVL R10,R4 ; Save original CNF address
0256'CF 9F 014D 786 PUSHAB W^185$ ; Push address of return
00000000'EF 16 0151 787 JSB NET$RESUME_NDI ; Call routine to build up stack, using
0157 788 ; the collating value
57 0C AC 7D 0157 789 MOVQ SAVREG(AP),R7 ; Restore R7, R8
53 24 A4 3C 015B 790 MOVZWL NDI ADD(R4),R3 ; Get the last node address
0A EF 015F 791 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number
53 53 06 0161 792 #TR4$S_ADDR_AREA,R3,R3 ;
54 54 DD 0164 793 PUSHL R4 ; Save old CNF address
54 00000000'EF DD 0166 794 MOVL NET$GL_PTR_VCB,R4 ; Get the RCB address
```

```
008B C4 53 91 016D 795 CMPB R3,RCB$B_HOMEAREA(R4) ; Is this in our area?
      42 13 0172 796 BEQL 90$ ; Br if yes, continue with our area
      SE 04 C0 0174 797 ADDL #4,SP ; Else, clean up stack
      07 50 E9 0177 798 BLBC R0,30$ ; Br if failure to proceed
      28 50 01 E0 017A 799 BBS #1,R0,60$ ; Br if take this one (next in tree)
      00D2 31 017E 800 BRW 180$ ; Else continue processing of tree
      0181 801 30$: ;
      0181 802 ; We could not continue with last node given, so we will
      0181 803 ; see if last was the DUM NDI and if so, try to continue anyway.
      0181 804 ;
      26 11 0181 805 BRB 80$ ; Should always leave now
      0183 806 ::88 CMPL R10,NET$GL_DUM_NDI ; Is this the dummy NDI?
      0183 807 ::88 BNEQ 80$ ; Br if not, no more NDIs
      0183 808 ::88 MOVL R11,R10 ; Start from beginning of list again
      0183 809 ::88 MOVL CNR$L_COLBTE(R11),R2 ; Get the collate tree root
      0183 810 ::88 BRB 110$ ; And start with our area
      0183 811 40$: ;
      0183 812 ; Engage in a co-routine dialogue with the user. The scanner half
      0183 813 ; of the dialogue owns the stack until the calling routine calls
      0183 814 ; back with any function code other than CNF$ADVANCE; the other
      0183 815 ; function codes causes the scanner to return to the caller with a
      0183 816 ; clean stack thus terminating the co-routine dialogue.
      0183 817 ;
      0183 818 ; Register usage:
      0183 819 ; R1 may be destroyed, by this routine, but must be preserved
      0183 820 ; on calls to the co-routine SCAN routine. Therefore, on initial
      0183 821 ; input R1 will be zero, but will be what the caller requires
      0183 822 ; on output. The same goes for R2.
      0183 823 ;
      04 AC 8E DO 0183 824 MOVL (SP)+,CALLER(AP) ; Save caller's return address
      51 OC AC 7D 0187 825 MOVQ SAVREG(AP),R1 ; Restore R1,R2
      14 AC 53 7D 018B 826 MOVQ R3,TEMPRG(AP) ; Save R3,R4
      54 00000000'EF DO 018F 827 MOVL NET$GL_PTR_VCB,R4 ; Get RCB pointer
      0196 828 $DISPATCH R0,- ; Dispatch on function code returned by
      0196 829 <- ; co-routine
      0196 830 <CNF$ADVANCE, 100$>,- ; Advance to next CNF, continue dialogue
      0196 831 <CNF$TAKE_PREV, 200$>,- ; Return previous CNF, abort dialogue
      0196 832 <CNF$QUIT, 300$>,- ; CNF not found, abort dialogue
      0196 833 <CNF$TAKE_CURR, 400$>,- ; Take current CNF, abort dialogue
      0196 834 > ;
      01A2 835 BUG_CHECK NETNOSTATE,FATAL
      01A6 836 ;
      50 01 DO 01A6 837 60$: MOVL #1,R0 ; Indicate success
      01A9 838 ;
      OC AC 51 7D 01A9 839 80$: MOVQ R1,SAVREG(AP) ; Save R1,R2
      53 14 AC 7D 01AD 840 MOVQ TEMPRG(AP),R3 ; Restore R3,R4
      04 BC 16 01B1 841 JSB @CALLER(AP) ; Call back the caller with status
      CD 11 01B4 842 BRB 40$ ;
      5A 8E DO 01B6 843 ;
      01B6 844 90$: MOVL (SP)+,R10 ; Restore CNF address
      01B9 845 ;
      01B9 846 100$: ;
      01B9 847 ; Advance to the next CNF.
      01B9 848 ;
      01B9 849 ;::88 NOTE that the following 2 instructions don't work too well when
      01B9 850 ;::88 R10 is pointing to the CNR.
      01B9 851 ;::88
```

```
1C AC 5A D0 01B9 852      MOVL R10,CNFADD(AP)      ; Save last CNF given back to caller
53 12 AA 3C 01BD 853      MOVZWL CNF$W_ID(R10),R3      ; Get current node address
20 AC 53 B0 01C1 854 110$: MOVW R3,NODADD(AP)      ; Save last node # given to caller
      0A EF 01C5 855 120$: EXTZV #TR4$V_ADDR_AREA,-      ; Get the area of the current node
50 53 06 91 01C7 856      #TR4$S_ADDR_AREA,R3,R0
008B C4 50 91 01CA 857      CMPB R0,RCB$B_HOMAREA(R4)      ; Is this our area?
      03 13 01CF 858      BEQL 140$      ; Br if yes, check all nodes in area
      007F 31 01D1 859      BRW 180$      ; Else, traverse the tree
      01D4 860      ;
      01D4 861      ; Process nodes in our area. Use NDIs if they exist, else
      01D4 862      ; use the DUM_NDI if the node is reachable.
      01D4 863      ;
      0E A4 53 B6 01D4 864 140$: INCW R3      ; Get next node address
      53 B1 01D6 865      CMPW R3,RCB$W_ADDR(R4)      ; Is this the local node?
      F8 13 01DA 866      BEQL 140$      ; Br if yes
      00 EF 01DC 867      EXTZV #TR4$V_ADDR_DEST,-      ; Get the node number within the area
50 53 0A 91 01DE 868      #TR4$S_ADDR_DEST,R3,R0
      4B 13 01E1 869      BEQL 150$      ; Br if wrap-around (max address = 1023)
5A A4 50 B1 01E3 870      CMPW R0,RCB$W_MAX_ADDR(R4)      ; Is the node still in our area?
      45 1A 01E7 871      BGTRU 150$      ; Br if not
      7E D4 01E9 872      CLRL -(SP)      ; Clean up stack
50 02 AE 53 90 01EB 873      MOVW R3,2(SP)      ; Stuff low byte of address
53 F8 8F 78 01EF 874      ASHL #8,R3,R0      ; Shift down the high byte
01 AE 50 90 01F4 875      MOVW R0,1(SP)      ; Stuff high byte of address
      57 03 9A 01F8 876      MOVZBL #3,R7      ; Set length of string
      58 SE D0 01FB 877      MOVL SP,R8      ; Point to string
      SA D4 01FE 878      CLRL R10      ; Start from beginning
      FDFA' 30 0200 879      PUSHQ R3      ; Save node address, RCB address
      5E 04 C0 0203 880      BSBW NET$FIND_NDI      ; Try to get the real NDI
      97 50 E8 0206 881      POPQ R3      ; Restore node address, RCB address
      00 EF 0209 882      ADDL #4,SP      ; Clean up stack
50 53 0A 91 020C 883      BLBS R0,60$      ; Call back caller with success
      1C B440 B5 020F 884      EXTZV #TR4$V_ADDR_DEST,-      ; Get the node number within the area
      BA 13 0211 885      #TR4$S_ADDR_DEST,R3,R0
      0214 886      TSTW @RCB$L_PTR_0A(R4)[R0]      ; Is it reachable?
      0218 887      BEQL 140$      ; If EQL then unreachable
      021A 888      ;
      021A 889      ; Point R10 to the dummy NDI, and set up the NDI fields
      021A 890      ;
5A 00000000'EF D0 021A 891      MOVL NET$GL_DUM_NDI,R10      ; Else, use the dummy NDI
      6A 7C 0221 892      CLRQ (R10)      ; Clear BTE pointers
      24 AA 53 B0 0223 893      MOVW R3,NDI_ADD(R10)      ; Stuff the address
      12 AA 53 B0 0227 894      MOVW R3,CNF$W_ID(R10)      ; Here too
      FF78 31 022B 895      BRW 60$      ; Call back caller with success
      022E 896      ;
      022E 897      ; We have exhausted all NDIs within our area, so we must now
      022E 898      ; find out where to pick up again in the collating tree. Skip
      022E 899      ; all NDIs in our area.
      022E 900      ;
      53 D7 022E 901      DECL R3      ; Back up in case we have full area
      52 D5 0230 902      TSTL R2      ; Was there a last node?
      35 13 0232 903      BEQL 190$      ; Br if not, failure
      0A EF 0234 904      EXTZV #TR4$V_ADDR_AREA,-      ; Get the area of the current node
53 53 06 91 0236 905      #TR4$S_ADDR_AREA,R3,R3
      08 BC 16 0239 906 160$: JSB @BTECOR(AP)      ; Skip to next NDI, call BTE co-routine
      2A 50 E9 023C 907      BLBC R0,190$      ; Br if failure, don't pop stack
      08 AC 8E D0 023F 908      MOVL (SP)+,BTECOR(AP)      ; Clean up stack
```

```
08 0B 04 E0 0243 909 BBS #NDI V LOOP - ; Br if this is a LOOP NODE
08 0B AA 0245 910 CNF$B FLG(R10),170$
0A ED 0248 911 CMPZV #TR4$V-ADDR-AREA,- ; Is this our area?
06 024A 912 #TR4$S-ADDR-AREA,-
53 12 AA 024B 913 CNF$W-ID(R10),R3
E9 15 024E 914 BLEQ 160$ ; Br if yes, skip this CNF
FF53 31 0250 915 170$: BRW 60$ ; Else, return the CNF
0253 916
0253 917 180$:
0253 918 ; Skip to next node in data base
0253 919
08 BC 16 0253 920 JSB @BTECOR(AP) ; Call back co-routine
10 50 E9 0256 921 185$: BLBC R0,190$ ; Br if failure, don't pop stack
08 AC 8E D0 0259 922 MOVL (SP)+,BTECOR(AP) ; Else, save return address
06 E1 025D 923 BBC #NDI V MARKER,- ; Never return the NDI marker
EE 0B AA 025F 924 CNF$B FLG(R10),170$
53 24 AA 3C 0262 925 MOVZWL NDI ADD(R10),R3 ; Else, get the node address
FF5C 31 0266 926 BRW 120$ ; Skip this CNF
0269 927
50 D4 0269 928 190$: CLRL R0 ; Say 'no more CNFs'
FF3B 31 026B 929 BRW 80$ ; Tell caller the bad news
026E 930
026E 931 200$:
026E 932 ; The caller wants to take the previous CNF
026E 933
5A 1C AC D0 026E 934 MOVL CNFADD(AP),R10 ; Get previous CNF address
17 13 0272 935 BEQL 300$ ; Branch if none
00000000'EF 5A D1 0274 936 CMPL R10,NET$GL_DUM_NDI ; Is this a phantom NDI (from vector)?
10 12 027B 937 BNEQ 400$ ; If not, go with it
50 20 AC B0 027D 938 MOVW NODADD(AP),R0 ; Get previous node address
24 AA 50 B0 0281 939 MOVW R0,NDI ADD(R10) ; Stuff the address
12 AA 50 B0 0285 940 MOVW R0,CNF$W-ID(R10) ; Here too
02 11 0289 941 BRB 400$ ; Take common exit
028B 942
028B 943 300$:
028B 944 ; The caller wants to quit
028B 945
5A D4 028B 946 CLRL R10 ; Nullify CNF pointer
028D 947
028D 948 ; The caller is done with the scan and wants the stack back.
028D 949
53 14 AC 7D 028D 950 400$: MOVQ TEMPRG(AP),R3 ; Restore R3,R4
50 04 AC D0 0291 951 MOVL CALLER(AP),R0 ; Get caller's return address
5E 5C D0 0295 952 MOVL AP,SP ; Restore original SP
5E 5C 8ED0 0298 953 POPL AP ; Restore AP
24 C0 029B 954 ADDL #9*4,SP ; Pop scratch storage
60 17 029E 955 JMP (R0) ; Return to caller
02A0 956
02A0 957 ;+
02A0 958 ; Get the collating value for this NDI, whether it's a real NDI or not.
02A0 959
02A0 960 : Inputs:
02A0 961 : R10 = NDI address (maybe DUMMY NDI)
02A0 962 : R7,R8 are scratch
02A0 963
02A0 964 : Outputs:
02A0 965 : R10 = NDI address
```

```
02A0 966 : R7,R8 descriptor for NDI collate string
02A0 967 :
02A0 968 : R3 is destroyed.
02A0 969 :-
02A0 970
02A0 971 GET_COLLATE:
02A0 972 MOVAB NDI_Z_COL,R8 ; Get address of collate buffer
02A0 973 CMPL R10,NET$GL_DUM_NDI ; Is this the dummy NDI?
02A0 974 BNEQ 50$ ; Br if no, get real collate value
02A0 975 CLRL (R8) ; Else, dummy up COLLATE VALUE
02A0 976 MOVAB NDI_ADD(R10),2(R8) ; Set low byte of node address
02A0 977 MOVAB NDI_ADD+1(R10),1(R8) ; Set high byte of node address
02A0 978 MOVL #3,R7 ; Set size of string
02A0 979 BRB 90$ ; Continue with collate value
02C1 980
02C1 981 50$: MOVL R8,R3 ; Copy output buffer address
02C4 982
02C4 983 PUSHR #*M<R0,R1,R2,R5,R6,R8,R9> ; Save registers
02C8 984 BSBW NET$NDI_S_COL ; Get the collating value
02CB 985 POPR #*M<R0,R1,R2,R5,R6,R8,R9> ; Restore registers
02CF 986
02CF 987 SUBL3 R8,R3,R7 ; Calculate size of collate string
02D3 988
02D3 989 90$: RSB
02D4 990
```

58 00000004'EF 9E 02A0 972
00000000'EF 5A D1 02A7 973
11 12 02AE 974
68 D4 02B0 975
02 A8 24 AA 90 02B2 976
01 A8 25 AA 90 02B7 977
57 03 D0 02BC 978
12 11 02BF 979
53 58 D0 02C1 981
0367 8F BB 02C4 983
0BF9 30 02C8 984
0367 8F BA 02CB 985
57 53 58 C3 02CF 987
05 02D3 989
02D4 990


```
02D4 992 .SBTTL NET$SCAN_AJI - SCAN AJI DATABASE
02D4 993 :+
02D4 994 : NET$SCAN_AJI - Scan AJI database
02D4 995 :
02D4 996 : This co-routine is used to scan the database, and return to the caller
02D4 997 : (co-routine) for each entry in the database. These routines establish
02D4 998 : the order of the database entries, above that of the natural ordering of
02D4 999 : the collating field.
02D4 1000 :
02D4 1001 : The search uses a dummy CNF which contains two pieces of information
02D4 1002 : (as well as supplying a valid CNF address): A identifier describing
02D4 1003 : the current adjacency being processed, and an identifier describing
02D4 1004 : the previous adjacency (so we can go backwards).
02D4 1005 :
02D4 1006 : Inputs:
02D4 1007 :
02D4 1008 : R11 = Address of CNR
02D4 1009 : R10 = Address of starting CNF (or 0 if to start at the beginning)
02D4 1010 :
02D4 1011 : Outputs:
02D4 1012 :
02D4 1013 : R10 = Address of CNF if dialogue aborted prematurely, else 0.
02D4 1014 :
02D4 1015 : The caller receives control on each database entry in list (via co-routine
02D4 1016 : call).
02D4 1017 :
02D4 1018 : On input to co-routine:
02D4 1019 :
02D4 1020 : R0 = True if entry was found. False if at end of list (R10 invalid)
02D4 1021 : R10 = Address of CNF entry found
02D4 1022 :
02D4 1023 : On output from co-routine:
02D4 1024 :
02D4 1025 : R0 = CNFS_ADVANCE Advance to next CNF, continue dialogue
02D4 1026 : CNFS_TAKE_PREV Return previous CNF, abort dialogue
02D4 1027 : CNFS_TAKE_CURR Return current CNF, abort dialogue
02D4 1028 : CNFS_QUIT Return no CNF (R10 = 0), abort dialogue
02D4 1029 :
02D4 1030 : *** These routines must be abortable via a RET ***
02D4 1031 : ---
02D4 1032 :
02D4 1033 : NET$SCAN_AJI::
02D4 1034 : TSTL R10 ; Adjacency scanner co-routine
02D4 1035 : BEQL 5$ ; Already pointing to a CNF ?
02D4 1036 : CMPL R11,R10 ; If EQL no, point to common CNF
02D4 1037 : BNEQ 10$ ; At head of list ?
02D4 1038 5$: MOVAB NET$T_CNF_AJI,R10 ; If NEQ no, assume R10 is valid
02D4 1039 : MOVW #LPD$C_LOC_INX,CNFSW_ID(R10) ; Point to internal dummy CNF
02D4 1040 : CLRW CNF$C_LENGTH(R10) ; Initialize search context
02D4 1041 10$: MOVL #1,R0 ; Initialize previous entry context
02D4 1042 20$: JSB @($P)+ ; Indicate success
02D4 1043 : $DISPATCH R0,<- ; Call back our caller
02D4 1044 :
02D4 1045 : <CNFS_ADVANCE, 30$> -; Advance to next CNF, continue dialogue
02D4 1046 : <CNFS_TAKE_PREV, 40$> -; Return previous CNF, abort dialogue
02D4 1047 : <CNFS_QUIT, 50$> -; CNF not found, abort dialogue
02D4 1048 : <CNFS_TAKE_CURR, 60$> -; Take current CNF, abort dialogue
```

SA 00000000'EF 9E 02D0 1038 5\$: MOVAB NET\$T_CNF_AJI,R10

12 AA 01 B0 02E4 1039 : MOVW #LPD\$C_LOC_INX,CNFSW_ID(R10)

24 AA B4 02E8 1040 : CLRW CNF\$C_LENGTH(R10)

50 01 D0 02EB 1041 10\$: MOVL #1,R0

9E 16 D0 02EE 1042 20\$: JSB @(\$P)+

```
02F0 1049 >
02FC 1050 BUG_CHECK NETNOSTATE,FATAL
0300 1051
12 AA B0 0300 1052 30$: MOVW CNF$W_ID(R10),- ; Save current position
24 AA 0303 1053 CNF$C_LENGTH(R10)
12 AA B6 0305 1054 35$: INCW CNF$W_ID(R10) ; Advance to next ADJ slot
58 12 AA 3C 0308 1055 MOVZWL CNF$W_ID(R10),R8 ; Get ADJ index
50 00000000'EF D0 030C 1056 MOVL NET$GC_PTR_VCB,R0 ; Get RCB address
68 A0 58 B1 0313 1057 CMPW R8,RCB$W_MAX_ADJ(R0) ; Within range?
0B 1A 0317 1058 BGTRU 38$ ; If not, terminate search
50 2C B048 D0 0319 1059 MOVL @RCB$C_PTR_ADJ(R0)[R8],R0 ; Get ADJ address
E3 60 00 E1 031E 1060 BBC #ADJ$V_INUSE,ADJ$B_STS(R0) 35$ ; If slot not in use, continue
C7 11 0322 1061 BRB 10$ ; Else, call back with success
50 D4 0324 1062 38$: CLRL R0 ; No more adjacencies
C6 11 0326 1063 BRB 20$ ; Call back with failure
0328 1064
24 AA B0 0328 1065 40$: MOVW CNF$C_LENGTH(R10),- ; Go back to previous link index
12 AA 032B 1066 CNF$W_ID(R10)
02 12 032D 1067 BNEQ 60$ ; If EQL then no previous link exists
5A D4 032F 1068 50$: CLRL R10 ; Nullify CNF pointer
05 0331 1069 60$: RSB ; Return to caller, terminate dialogue
```

```
0332 1071 .SBTTL NET$SCAN_SDI - SCAN SDI DATABASE
0332 1072 :+
0332 1073 : NET$SCAN_SDI - Scan SDI database
0332 1074 :
0332 1075 : This co-routine is used to scan the database, and return to the caller
0332 1076 : (co-routine) for each entry in the database. These routines establish
0332 1077 : the order of the database entries, above that of the natural ordering of
0332 1078 : the collating field.
0332 1079 :
0332 1080 : Each entry in this database corresponds to a DWB in the global linked
0332 1081 : list of DWBs. Each DWB represents a DLE session between a MOM process
0332 1082 : and a remote node.
0332 1083 :
0332 1084 : The search uses a dummy CNF which contains two pieces of information
0332 1085 : (as well as supplying a valid CNF address): An identifier describing
0332 1086 : the current DWB being processed, and an identifier describing
0332 1087 : the previous DWB (so we can go backwards).
0332 1088 :
0332 1089 : The database is collated on a identifier in the DWB which is unique
0332 1090 : among all the DWBs in the system.
0332 1091 :
0332 1092 : Inputs:
0332 1093 :
0332 1094 :     R11 = Address of CNR
0332 1095 :     R10 = Address of starting CNF (or 0 if to start at the beginning)
0332 1096 :
0332 1097 : Outputs:
0332 1098 :
0332 1099 :     R10 = Address of CNF if dialogue aborted prematurely, else 0.
0332 1100 :
0332 1101 : The caller receives control on each database entry in list (via co-routine
0332 1102 : call).
0332 1103 :
0332 1104 : On input to co-routine:
0332 1105 :
0332 1106 :     R0 = True if entry was found. False if at end of list (R10 invalid)
0332 1107 :     R10 = Address of CNF entry found
0332 1108 :
0332 1109 : On output from co-routine:
0332 1110 :
0332 1111 :     R0 = CNFS_ADVANCE      Advance to next CNF, continue dialogue
0332 1112 :         CNFS_TAKE_PREV    Return previous CNF, abort dialogue
0332 1113 :         CNFS_TAKE_CURR    Return current CNF, abort dialogue
0332 1114 :         CNFS_QUIT         Return no CNF (R10 = 0), abort dialogue
0332 1115 :
0332 1116 : *** These routines must be abortable via a RET ***
0332 1117 : ---
0332 1118 :
0332 1119 NET$SCAN_SDI::
0332 1120 TSTL R10 ; DLE scanner co-routine
0332 1121 BEQL 5$ ; Already pointing to a CNF ?
0332 1122 CMPL R11,R10 ; If EQL no, point to common CNF
0332 1123 BNEQ 10$ ; At head of list ?
0332 1124 5$: MOVAB NET$T_CNF_SDI,R10 ; If NEQ no, assume R10 is valid
0332 1125 CLRW CNFSW_ID(R10) ; Point to internal dummy CNF
0332 1126 CLRW CNFSC_LENGTH(R10) ; Initialize search context
0332 1127 10$: MOVL #1,R0 ; Initialize previous entry context
0332 1128 ; Indicate success
```

SA	05	D5	0332	1120	TSTL	R10			
	05	13	0334	1121	BEQL	5\$			
SA	5B	D1	0336	1122	CMPL	R11,R10			
	0D	12	0339	1123	BNEQ	10\$			
SA	00000000	EF	9E	033B	1124	5\$: MOVAB	NET\$T_CNF_SDI,R10		
	12	AA	B4	0342	1125	CLRW	CNFSW_ID(R10)		
	24	AA	B4	0345	1126	CLRW	CNFSC_LENGTH(R10)		
50	01	D0	0348	1127	10\$: MOVL	#1,R0			

```
9E 16 034B 1128 20$: JSB @ (SP)+ ; Call back our caller
034D 1129 $DISPATCH R0,<-
034D 1130
034D 1131 <CNFS_ADVANCE, 30$> -; Advance to next CNF, continue dialogue
034D 1132 <CNFS_TAKE_PREV, 40$> -; Return previous CNF, abort dialogue
034D 1133 <CNFS_QUIT, 50$> -; CNF not found, abort dialogue
034D 1134 <CNFS_TAKE_CURR, 60$> -; Take current CNF, abort dialogue
034D 1135
0359 1136 >
035D 1137 BUG_CHECK NETNOSTATE,FATAL
12 AA 80 035D 1138 30$: MOVW CNFSW_ID(R10),- ; Save current position
24 AA 0360 1139 CNF$C_LENGTH(R10)
51 01 D0 0362 1140 MOVL #1,R1 ; Indicate that we want the 'next one'
1A 10 0365 1141 BSBB GET_DWB ; Locate next DWB
26 AA 51 D0 0367 1142 MOVL R1,CNF$C_LENGTH+2(R10) ; Store address of DWB for action routines
DE 11 0368 1143 BRB 20$ ; Return to caller with status
036D 1144
24 AA 80 036D 1145 40$: MOVW CNF$C_LENGTH(R10),- ; Go back to previous link index
12 AA 0370 1146 CNFSW_ID(R10)
0A 13 0372 1147 BEQL 50$ ; If EQL then no previous link exists
51 D4 0374 1148 CLRL R1 ; Indicate that we want 'this one'
09 10 0376 1149 BSBB GET_DWB ; Locate next DWB
26 AA 51 D0 0378 1150 MOVL R1,CNF$C_LENGTH+2(R10) ; Store address of DWB for action routines
02 11 037C 1151 BRB 60$ ; Abort dialogue, return with CNF
037E 1152
5A D4 037E 1153 50$: CLRL R10 ; Nullify CNF pointer
05 0380 1154 60$: RSB ; Return to caller, terminate dialogue
0381 1155
0381 1156 ;
0381 1157 ; Local subroutine to locate a DWB in the global DWB list given it's
0381 1158 ; unique NETACP channel number.
0381 1159 ;
0381 1160
0381 1161 GET_DWB:
0381 1162 PUSHF #M<R2,R3,R4,R5> ; Save registers
0383 1163 MOVL NET$GL_DLE_UCB0,R0 ; Get ND's UCB0
038A 1164 MOVAB UCBSQ_DWB_LIST(R0),R2 ; Get address of listead
038F 1165 MOVL R2,R5 ; Setup for loop
0392 1166 10$: MOVL (R5),R5 ; Get next entry
0395 1167 CMPL R5,R2 ; End of list?
0398 1168 BEQL 50$ ; Branch if so
039A 1169 CMPW DWBSW_ID(R5),- ; Have we found the right spot?
039D 1170 CNFSW_ID(R10)
039F 1171 BLSSU 10$ ; If still LSS, keep scanning
03A1 1172 BEQL 15$ ; Branch if same
03A3 1173 ;
03A3 1174 ; We have found the 'next entry' in the collating sequence,
03A3 1175 ; if the current one went away.
03A3 1176 ;
1A 51 E9 03A3 1177 BLBC R1,50$ ; If caller wanted current,
0B 11 03A6 1178 BRB 20$ ; then too bad, else use next
03A8 1179 ;
03A8 1180 ; We have found the 'current entry' in the collating sequence.
03A8 1181 ;
08 51 E9 03A8 1182 15$: BLBC R1,20$ ; If caller wanted next one,
55 65 D0 03AB 1183 MOVL (R5),R5 ; Skip to next one
52 55 D1 03AE 1184 CMPL R5,R2 ; End of list?
```

50	0D	13	03B1	1185	BEQL	50\$	; If so, report error
51	01	D0	03B3	1186	MOVL	#1,R0	; Success
	55	D0	03B6	1187	MOVL	R5,R1	; Return DWB address in R1
4E	A5	B0	03B9	1188	MOVW	DWB\$W_ID(R5),-	; Stuff the collating value in
12	AA		03BC	1189		CNF\$W_ID(R10)	; the CNF
	04	11	03BE	1190	BRB	90\$	
	50	D4	03C0	1191	CLRL	R0	; Failure
	51	D4	03C2	1192	CLRL	R1	; Make sure DWB address is 0
	3C	BA	03C4	1193	POPR	#*M<R2,R3,R4,R5>	; Restore registers
	05	03C6	1194	RSB			

```
03C7 1196 .SBTTL NET$SCAN_ARI - SCAN ARI DATABASE
03C7 1197 :+
03C7 1198 : NET$SCAN_ARI - Scan ARI (area) database
03C7 1199 :
03C7 1200 : This co-routine is used to scan the database, and return to the caller
03C7 1201 : (co-routine) for each entry in the database. These routines establish
03C7 1202 : the order of the database entries, above that of the natural ordering of
03C7 1203 : the collating field.
03C7 1204 :
03C7 1205 : The search uses a dummy CNF which contains two pieces of information
03C7 1206 : (as well as supplying a valid CNF address): A identifier describing
03C7 1207 : the current area being processed, and an identifier describing
03C7 1208 : the previous area (so we can go backwards).
03C7 1209 :
03C7 1210 : Inputs:
03C7 1211 :
03C7 1212 :     R11 = Address of CNR
03C7 1213 :     R10 = Address of starting CNF (or 0 if to start at the beginning)
03C7 1214 :
03C7 1215 : Outputs:
03C7 1216 :
03C7 1217 :     R10 = Address of CNF if dialogue aborted prematurely, else 0.
03C7 1218 :
03C7 1219 : The caller receives control on each database entry in list (via co-routine
03C7 1220 : call).
03C7 1221 :
03C7 1222 : On input to co-routine:
03C7 1223 :
03C7 1224 :     R0 = True if entry was found. False if at end of list (R10 invalid)
03C7 1225 :     R10 = Address of CNF entry found
03C7 1226 :
03C7 1227 : On output from co-routine:
03C7 1228 :
03C7 1229 :     R0 = CNFS_ADVANCE      Advance to next CNF, continue dialogue
03C7 1230 :         CNFS_TAKE_PREV    Return previous CNF, abort dialogue
03C7 1231 :         CNFS_TAKE_CURR    Return current CNF, abort dialogue
03C7 1232 :         CNFS_QUIT         Return no CNF (R10 = 0), abort dialogue
03C7 1233 :
03C7 1234 : *** These routines must be abortable via a RET ***
03C7 1235 : ---
03C7 1236 :
03C7 1237 NET$SCAN_ARI::
03C7 1238 TSTL R10 ; Area scanner co-routine
03C7 1239 BEQL 5$ ; Already pointing to a CNF ?
03C7 1240 CMPL R11,R10 ; If EQL no, point to common CNF
03C7 1241 BNEQ 10$ ; At head of list ?
03C7 1242 5$: MOVAB NET$T_CNF_ARI,R10 ; If NEQ no, assume R10 is valid
03C7 1243 CLRW CNFSW_ID(R10) ; Point to internal dummy CNF
03C7 1244 CLRW CNF$C_LENGTH(R10) ; Initialize search context
03C7 1245 10$: MOVL #1,R0 ; Initialize previous entry context
03C7 1246 20$: JSB @($P)+ ; Indicate success
03C7 1247 $DISPATCH R0,<- ; Call back our caller
03C7 1248
03C7 1249 <CNFS_ADVANCE, 30$> -; Advance to next CNF, continue dialogue
03C7 1250 <CNFS_TAKE_PREV, 40$> -; Return previous CNF, abort dialogue
03C7 1251 <CNFS_QUIT, 50$> -; CNF not found, abort dialogue
03C7 1252 <CNFS_TAKE_CURR, 60$> -; Take current CNF, abort dialogue
```

SA	00000000	EF	9E	03D0	1242	5\$:	MOVAB	NET\$T_CNF_ARI,R10
	12	AA	B4	03D7	1243		CLRW	CNFSW_ID(R10)
	24	AA	B4	03DA	1244		CLRW	CNF\$C_LENGTH(R10)
	50	01	D0	03DD	1245	10\$:	MOVL	#1,R0
		9E	16	03E0	1246	20\$:	JSB	@(\$P)+
				03E2	1247		\$DISPATCH	R0,<-
				03E2	1248			
				03E2	1249		<CNFS_ADVANCE,	30\$>
				03E2	1250		<CNFS_TAKE_PREV,	40\$>
				03E2	1251		<CNFS_QUIT,	50\$>
				03E2	1252		<CNFS_TAKE_CURR,	60\$>

```

      03E2 1253      >
      03EE 1254      BUG_CHECK      NETNOSTATE,FATAL
      03F2 1255
12 AA B0 03F2 1256 30$: MOVW      CNF$W_ID(R10),-      ; Save current position
24 AA      03F5 1257      CNF$C_LENGTH(R10)
12 AA B6 03F7 1258 35$: INCW      CNF$W_ID(R10)      ; Advance to next area number
58 12 AA 3C 03FA 1259      CNF$W_ID(R10),R8      ; Get area number
50 00000000'EF D0 03FE 1260      MOVZWL CNF$W_ID(R10),R8      ; Get RCB address
008B C0 58 91 0405 1261      MOVL      NET$G[ PTR VCB,R0      ; Get RCB address
      D1 13 040A 1262      CMPB      R8,RCB$B_HOMEAREA(R0)      ; Our own area?
008C C0 58 B1 040C 1263      BEQL      10$      ; If so, return it
      OD 1A 0411 1264      CMPW      R8,RCB$B_MAX_AREA(R0)      ; Within range?
20 A0 D5 0413 1265      BGTRU      38$      ; If not, terminate search
      DF 13 0416 1266      TSTL      RCB$B_PTR_AOA(R0)      ; Are we a level 2 router?
20 B048 B5 0418 1267      BEQL      35$      ; If not, skip the following
      D9 13 041C 1268      TSTW      @RCB$B_PTR_AOA(R0)[R8]      ; Is area reachable?
      BD 11 041E 1269      BEQL      35$      ; If not, then skip to next one
50 D4 0420 1270 38$: BRB      10$      ; Else, return it
      BC 11 0422 1271      CLRL      R0      ; No more adjacencies
      0424 1272      BRB      20$      ; Call back with failure
24 AA B0 0424 1273 40$: MOVW      CNF$C_LENGTH(R10),-      ; Go back to previous entry
12 AA      0427 1274      CNF$W_ID(R10)
      02 12 0429 1275      BNEQ      60$      ; If EQL then no previous entry exists
5A D4 042B 1276 50$: CLRL      R10      ; Nullify CNF pointer
      05 042D 1277 60$: RSB      ; Return to caller, terminate dialogue
```

```
042E 1279 .SBTTL NET$SPCSCAN_XXX - SPECIAL DATABASE SCAN ROUTINES
042E 1280 :+
042E 1281 : NET$SPCSCAN_XXX - Special database scan routines
042E 1282 :
042E 1283 : These routines are called to scan a CNF entry if the search operator is EQL.
042E 1284 :
042E 1285 : Inputs:
042E 1286 :
042E 1287 :     R11 = Address of CNR
042E 1288 :     R10 = Address of CNF
042E 1289 :     R9  = Field ID of search field.
042E 1290 :     R7,R8 = Descriptor of search key value
042E 1291 :
042E 1292 : Outputs:
042E 1293 :
042E 1294 :     R0 = Always clear.
042E 1295 :         Bit 0: Set if success, else clear.
042E 1296 :         Bit 1: Set if key is recognized, else clear.
042E 1297 :
042E 1298 :-
042E 1299 :
042E 1300 NET$SPCSCAN_CRI:: ; Circuit CNF real SCAN routine
042E 1301 NET$SPCSCAN_PLI:: ; Line CNF real SCAN routine
042E 1302 NET$SPCSCAN_LNI:: ; Local node CNF real SCAN routine
042E 1303 NET$SPCSCAN_OBI:: ; Object CNF scanner SCAN routine
042E 1304 NET$SPCSCAN_EFI:: ; Event filter CNF SCAN routine
042E 1305 NET$SPCSCAN_ESI:: ; Event sink CNF SCAN routine
042E 1306 NET$SPCSCAN_SPI:: ; Server process CNF SCAN routine
042E 1307 NET$SPCSCAN_LLI:: ; Logical link CNF SCAN routine
042E 1308 NET$SPCSCAN_AJI:: ; Adjacency CNF SCAN routine
042E 1309 NET$SPCSCAN_SDI:: ; DLE CNF SCAN routine
042E 1310 NET$SPCSCAN_ARI:: ; AREA Adjacency CNF SCAN routine
50  D4 042E 1311 CLRE R0
    05 0430 1312 RSB ; Always return false
```



```
0431 1314 .SBTTL NET$SPCSCAN_NDI - SPECIAL SCAN OF NDI DATABASE
0431 1315 :+
0431 1316 : NET$SPCSCAN_NDI - Special scan of NDI database
0431 1317 :
0431 1318 : This routine is used to scan the NDI database and return to the caller
0431 1319 : the address of the CNF. This routine can only be called if the operation
0431 1320 : is an EQL or FNDPOS operation. This routine will return any NDI except
0431 1321 : for the MARKER NDI.
0431 1322 :
0431 1323 : Inputs:
0431 1324 :
0431 1325 : R11 = Address of CNR
0431 1326 : R10 = Address of starting CNF (or 0 if to start at the beginning)
0431 1327 : R9 = Field ID of search field.
0431 1328 : R7,R8 = Descriptor of search key or value
0431 1329 :
0431 1330 : Outputs:
0431 1331 :
0431 1332 : R10 = Address of CNF if success, else 0.
0431 1333 : R0 = Bit 0: Set if success, else clear.
0431 1334 : Bit 1: Set if key is recognized, else clear.
0431 1335 :
0431 1336 :--
0431 1337 :
0431 1338 NET$SPCSCAN_NDI::
0431 1339 PUSHR #^M<R3,R4,R7,R8> : Find NDI block
0431 1340 CLRL R0 : Save registers
0431 1341 CMPL R9,#NFB$C_NDI_COL : Assume failure
0431 1342 BEQL 20$ : Searching by collating value?
0431 1343 CMPL R9,#NFB$C_NDI_ADD : If so, take it
0431 1344 BEQL 50$ : Searching by node address?
0431 1345 CMPL R9,#NFB$C_NDI_TAD : If so, take it
0431 1346 BEQL 30$ : Searching by transformed node address?
0431 1347 CMPL R9,#NFB$C_NDI_NNA : If so, take it
0431 1348 BEQL 10$ : Searching by node name?
0431 1349 BRW 110$ : If so, take it
0431 1350 : Else, leave with no match
0431 1351 :
0431 1352 : Name value, search the NAME tree for match
0431 1353 10$: BSBW NET$FIND_NAME : Search the name tree
0431 1354 BRW 100$ : Exit
0431 1355 :
0431 1356 : Collate value, search the COLLATE tree for match
0431 1357 :
0431 1358 20$: BSBW NET$FIND_NDI : Search the collate tree
0431 1359 BRW 100$ : Exit
0431 1360 :
0431 1361 : Transformed node address
0431 1362 :
0431 1363 : If the node address equals the executors node address
0431 1364 : then change it to zero;
0431 1365 : Swap the bytes of the node address field
0431 1366 :
0431 1367 30$: MOVL NET$GL_PTR_VCB,R0 : Get the RCB address
0431 1368 CMPW R8,RCB$W_ADDR(R0) : Is this the executors address?
0431 1369 BNEQ 50$ : Br if no, then okay
0431 1370 CLRL R8 : Else, zero the node address
```

0198 8F BB 0431 1339
50 D4 0435 1340
02020040 8F 59 D1 0437 1341
24 13 043E 1342
02010012 8F 59 D1 0440 1343
30 13 0447 1344
02010010 8F 59 D1 0449 1345
18 13 0450 1346
02020043 8F 59 D1 0452 1347
03 13 0459 1348
009B 31 045B 1349
045E 1350
045E 1351
045E 1352
FB9F' 30 045E 1353 10\$: BSBW NET\$FIND_NAME
0092 31 0461 1354 BRW 100\$
0464 1355
0464 1356
0464 1357
FB99' 30 0464 1358 20\$: BSBW NET\$FIND_NDI
008C 31 0467 1359 BRW 100\$
046A 1360
046A 1361
046A 1362
046A 1363
046A 1364
046A 1365
046A 1366
50 00000000'EF D0 046A 1367 30\$: MOVL NET\$GL_PTR_VCB,R0
OE AO 58 B1 0471 1368 CMPW R8,RCB\$W_ADDR(R0)
02 12 0475 1369 BNEQ 50\$
58 D4 0477 1370 CLRL R8

```

                                0479 1371
                                0479 1372
                                0479 1373
                                ; Node address
                                0479 1374 50$: PUSHL R8
                                047B 1375 CLRL -(SP)
                                047D 1376 MOVB R8,2(SP)
50 02 AE 58 90 0481 1377 ASHL #-8,R8,R0
58 F8 8F 78 0486 1378 MOVB R0,1(SP)
01 AE 50 90 048A 1379 MOVZBL #3,R7
57 03 9A 048D 1380 MOVL SP,R8
58 5E D0 0490 1381 BSBW NET$FIND_NDI
FB6D' 30 0493 1382 ADDL #4,SP
5E 04 C0 0496 1383 POPL R8
8ED0 0499 1384 BLBS R0,80$
51 50 E8 049C 1385 BBS #NET$V_INTRNL,-
09 E0 049E 1386 NET$GL_FLAGS,100$
52 00000000'EF 04A4 1387
                                5A D5 04A4 1388 TSTL R10
                                05 13 04A6 1389 BEQL 60$
                                5B 5A D1 04A8 1390 CMPL R10,R11
                                49 12 04AB 1391 BNEQ 100$
54 00000000'EF D0 04AD 1392 60$: MOVL NET$GL_PTR_VCB,R4
0A EF 04B4 1393 EXTZV #TR4$V-ADDR-AREA,-
53 58 06 04B6 1394 #TR4$S-ADDR-AREA,R8,R3
008B C4 53 91 04B9 1395 CMPB R3,RCB$B_HOMEAREA(R4)
36 12 04BE 1396 BNEQ 100$
0E A4 58 B1 04C0 1397 CMPW R8,RCB$W_ADDR(R4)
30 13 04C4 1398 BEQL 100$
00 EF 04C6 1399 EXTZV #TR4$V-ADDR_DEST,-
53 58 0A 04C8 1400 #TR4$S-ADDR_DEST,R8,R3
29 13 04CB 1401 BEQL 100$
5A A4 53 B1 04CD 1402 CMPW R3,RCB$W_MAX_ADDR(R4)
23 1A 04D1 1403 BGTRU 100$
1C B443 B5 04D3 1404 TSTW @RCB$L_PTR_OA(R4)[R3]
1D 13 04D7 1405 BEQL 100$
5A 00000000'EF D0 04D9 1406 MOVL NET$GL_DUM_NDI,R10
12 AA 58 B0 04E0 1407 MOVW R8,CNF$W_ID(R10)
24 AA 58 B0 04E4 1408 MOVW R8,NDI_ADD(R10)
50 01 9A 04E8 1409 MOVZBL #1,R0
09 11 04EB 1410 BRB 100$
06 E1 04ED 1411 80$: BBC #NDI_V_MARKER,-
04 0B AA 04EF 1412 CNF$B_FLG(R10),100$
50 D4 04F2 1413 90$: CLRL R0
5A D4 04F4 1414 CLRL R10
50 02 88 04F6 1415 100$: BISB #2,R0
0198 8F BA 04F9 1416 110$: POPR #^M<R3,R4,R7,R8>
05 04FD 1417 RSB
                                ; Save node address
                                ; Make some room on the stack
                                ; Store low byte
                                ; Swap down high byte
                                ; Store high byte
                                ; Set string length
                                ; Point R8 to string
                                ; Search the collate tree
                                ; Cleanup stack
                                ; Restore node address
                                ; Br if success
                                ; If internal, then not interested
                                ; in 'phantom' NDIs
                                ; Was starting CNF zero?
                                ; Br if yes, okay
                                ; Else, is starting CNF the CNR?
                                ; Br if not, return error
                                ; Get the RCB address
                                ; Get the area address
                                ; Is this our area?
                                ; Br if not, return failure
                                ; Is this for the local node?
                                ; Br if yes, return error
                                ; Get the node address
                                ; Br if zero, return error
                                ; Is it still in the area?
                                ; Br if not, return error
                                ; Is the node reachable?
                                ; Br if not, return error
                                ; Else, use the dummy NDI
                                ; Stuff the node address
                                ; ..here also
                                ; Return success
                                ; Exit
                                ; Never return the marker CNF
                                ; Indicate failure
                                ; Return error
                                ; Return indicator that key accepted
                                ; Restore registers
                                ; Return to caller
```

```
04FE 1419 .SBTTL NET$PRE_QIO_XXX - PRE-QIO PROCESSING
04FE 1420 :+
04FE 1421 : NET$PRE_QIO_XXX - Perform pre-QIO database processing
04FE 1422 :
04FE 1423 : This routine is called just after validating the NFB for a database
04FE 1424 : function to do any special pre-processing before the request is attempted.
04FE 1425 :
04FE 1426 : Inputs:
04FE 1427 :
04FE 1428 : R11 = Address of CNR
04FE 1429 : NET$GQ_SRCH_KEY = Descriptor of search key value
04FE 1430 : NET$GL_SRCH_ID = Field ID of search field.
04FE 1431 :
04FE 1432 : Outputs:
04FE 1433 :
04FE 1434 : NET$GQ_SRCH_KEY = New search key value (if reformatted)
04FE 1435 : ---
04FE 1436 : NET$PRE_QIO_NDI::
02010010 8F 00000000'EF D1 04FE 1437 CMPL NET$GL_SRCH_ID,#NFB$C_NDI_TAD ; Searching by node address?
04FE 1438 BEQL 10$ ; If so, transform key
02010012 8F 00000000'EF D1 050B 1439 CMPL NET$GL_SRCH_ID,#NFB$C_NDI_ADD ; Searching by node address?
04FE 1440 BNEQ 90$ ; If not, skip it
51 00J00004'EF D0 0518 1441 10$: MOVL NET$GQ_SRCH_KEY+4,R1 ; Get the node address
04FE 1442 BEQL 90$ ; If 0, skip it
04FE 1443 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number
50 51 06 0523 1444 #TR4$S_ADDR_AREA,R1,R0
04FE 1445 BNEQ 90$ ; Check it if non-zero
52 00000000'EF D0 0528 1446 MOVL NET$GL_PTR_VCB,R2 ; Get RCB address
04FE 1447 INSV RCB$B #OME$AREA(R2),- ; If area = 0, then use our area
04FE 1448 #TR4$V_ADDR_AREA,- ; (accept 0 as synonym for our area)
51 06 0534 1449 #TR4$S_ADDR_AREA,R1
04FE 1450 BSBW SUPPRESS_AREA ; Suppress area, if necessary, to be
04FE 1451 ; consistent with the TAD which is also
04FE 1452 ; suppressed (so that it will match)
00000004'EF 51 D0 0539 1453 MOVL R1,NET$GQ_SRCH_KEY+4 ; Set new search key
04FE 1454 90$:
04FE 1455 NET$PRE_QIO_LNI::
04FE 1456 NET$PRE_QIO_OBI::
04FE 1457 NET$PRE_QIO_EFI::
04FE 1458 NET$PRE_QIO_ESI::
04FE 1459 NET$PRE_QIO_LLI::
04FE 1460 NET$PRE_QIO_SPI::
04FE 1461 NET$PRE_QIO_AJI::
04FE 1462 NET$PRE_QIO_SDI::
04FE 1463 NET$PRE_QIO_ARI::
50 00' D0 0540 1464 MOVL S^#SS$_NORMAL,R0 ; Indicate success
05 0543 1465 RSB
```

```

0544 1467 .SBTTL NET$SHOW_XXX - PRE-SHOW PROCESSING
0544 1468 :+
0544 1469 : NET$SHOW_XXX - Show QIO pre-processing
0544 1470 :
0544 1471 : This routine is called for each CNF which is about to be returned
0544 1472 : to a 'show' QIO.
0544 1473 :
0544 1474 : Inputs:
0544 1475 :
0544 1476 : R11 = Address of CNR
0544 1477 : R10 = Address of CNF
0544 1478 :
0544 1479 : Outputs:
0544 1480 :
0544 1481 : R0 = Status code
0544 1482 :
0544 1483 : The CNF may be updated.
0544 1484 :-
0544 1485 :
0544 1486 NET$SHOW_LNI::
0544 1487 NET$SHOW_NDI::
0544 1488 NET$SHOW_OBI::
0544 1489 NET$SHOW_EFI::
0544 1490 NET$SHOW_ESI::
0544 1491 NET$SHOW_LLI::
0544 1492 NET$SHOW_SPI::
0544 1493 NET$SHOW_AJI::
0544 1494 NET$SHOW_SDI::
0544 1495 NET$SHOW_ARI::
50 00' D0 0544 1496 MOVL S^#SS$_NORMAL,R0 ; Indicate success
0547 1497 RSB

```

```
0548 1499 .SBTTL NET$DEFAULT_xxx - APPLY DEFAULT VALUES
0548 1500 :+
0548 1501 : NET$DEFAULT_xxx - Apply default values to selected CNF parameters.
0548 1502 :
0548 1503 : This routine is called by CNF$INSERT just prior to validating a CNF
0548 1504 : entry which is to be inserted into the database. Its purpose is to
0548 1505 : supply default values to selected parameters.
0548 1506 :
0548 1507 : INPUTS:      R11      CNR pointer
0548 1508 :              R10      CNF pointer
0548 1509 :
0548 1510 : OUTPUTS:     R11      CNR pointer
0548 1511 :              R10      CNF pointer
0548 1512 :              R0       $$$_NORMAL -- this routine always succeeds.
0548 1513 :
0548 1514 : All other registers contain garbage.
0548 1515 :-
0548 1516 NET$DEFAULT_LNI::
0548 1517 $GETFLD lni,l,ety ; Get executor type
0548 1518 BLBS RO,NET$APPLY_DFLT ; If specified, then continue
0548 1519 JSB NET$GET_END ; Get endnode info
0548 1520 BLBC RO,NET$APPLY_DFLT ; If failure, then use default
0548 1521 MOVL #ADJ$C_PTY_PR4N,R8 ; Use endnode as default
0548 1522 BSBW CNF$PUT_FIELD ; Store it in the CNF
0567 1523 NET$DEFAULT_OBI::
0567 1524 NET$DEFAULT_EFI::
0567 1525 NET$DEFAULT_ESI::
0567 1526 NET$DEFAULT_LLI::
0567 1527 NET$DEFAULT_SPI::
0567 1528 NET$DEFAULT_AJI::
0567 1529 NET$DEFAULT_SDI::
0567 1530 NET$DEFAULT_ARI::
0567 1531 NET$APPLY_DFLT::
0567 1532 MOVZBL CNR$B_TYPE(R11),R0 ; Apply default CNF parameter values
0567 1533 MOVL NET$AC_CNF_DFLT[R0],R6 ; Get database i.d.
0573 1534 ; Get parameter id,value table
0573 1535 NET$TABLE_DFLT::
0573 1536 ; Apply defaults given table address
0573 1537 10$: MOVL (R6)+,R9 ; R6 = default table address
0576 1538 BEQL 50$ ; Get parameter i.d., advance R6
0578 1539 BSBW CNF$GET_FIELD ; Done if EQL
0578 1540 MOVL (R6)+,R8 ; See if field is already setup
0578 1541 BLBS RO,10$ ; Get parameter value, advance R6
0581 1542 BSBW CNF$PUT_FIELD ; If LBS the no need for default
0584 1543 BRB 10$ ; Store it in the CNF
0586 1544 50$: MOVL S^#$$$_NORMAL,R0 ; Ignore errors
0589 1545 RSB ; Always successful
; Done
```

```
058A 1547 .SBTTL NET$DEFAULT_NDI - APPLY DEFAULT VALUES TO NDI CNF
058A 1548 :+
058A 1549 : NET$DEFAULT_NDI - Apply default values to NDI CNF parameters -
058A 1550 : also set the LOOP bit in the FLG byte if needed.
058A 1551 :
058A 1552 : This routine is called by CNF$INSERT just prior to validating a CNF
058A 1553 : entry which is to be inserted into the database. Its purpose is to
058A 1554 : supply default values to selected parameters.
058A 1555 :
058A 1556 : INPUTS:      R11      CNR pointer
058A 1557 :           R10      CNF pointer
058A 1558 :
058A 1559 : OUTPUTS:     R11      CNR pointer
058A 1560 :           R10      CNF pointer
058A 1561 :           R0       SS$_NORMAL -- this routine always succeeds.
058A 1562 :
058A 1563 : All other registers contain garbage.
058A 1564 :-
058A 1565 NET$DEFAULT_NDI::
DB 10 058A 1566 BSBB NET$APPLY_DFLT ; Apply defaults
18 50 E9 058C 1567 $GETFLD ndi,l,add ; Get the node address
58 D5 0599 1568 BLBC R0,90$ ; Br if none, will fail later
14 12 059C 1569 TSTL R8 ; Is node address zero?
05A0 1570 BNEQ 90$ ; Br if no, not a loop node
04 50 E9 05AD 1571 $GETFLD ndi,s,nli ; Get the optional circuit name
10 88 05BD 1572 BLBC R0,90$ ; Br if none, not a loop node
08 AA 05B0 1573 BISB #1&NDI_V_LOOP,- ; Mark this as a 'LOOP' node
50 00' D0 05B4 1574 CNF$B_FLG(R10)
05 05B7 1575 90$: MOVL S^SS$_NORMAL,R0 ; Always successful
RSB ; Done
```

```
05B8 1578 .SBTTL NET$INSERT_LNI - PRE-INSERTION PROCESSING
05B8 1579 :+
05B8 1580 : NET$INSERT_LNI - Insert an LNI entry into database
05B8 1581 :
05B8 1582 : This routine is called to validate the CNF entry before inserting
05B8 1583 : it into the database.
05B8 1584 :
05B8 1585 : Inputs:
05B8 1586 :
05B8 1587 : R11 = Address of CNR
05B8 1588 : R10 = Address of CNF
05B8 1589 :
05B8 1590 : Outputs:
05B8 1591 :
05B8 1592 : R0 = Status code. If error, entry is not inserted.
05B8 1593 :-
05B8 1594
05B8 1595 NET$INSERT_LNI::
05B8 1596 $GETFLD lni,l,add ; Get the node address
05C5 1597 BLBS R0,5$ ; If LBC then report bad node address
05C8 1598 BRW 110$ ;
05CB 1599 5$: MOVL R8,R5 ; Has the address been set yet?
05CE 1600 BEQL 10$ ; If not set, skip address checks
05D0 1601 MOVL NET$GL_PTR_VCB,R4 ; Get RCB address
05D7 1602 CMPW R8,RCB$W_ADDR(R4) ; Is the address being changed?
05DB 1603 BNEQ 20$ ; If so, perform address checks
05DD 1604 10$: BRW 140$ ; Skip all address checks
05E0 1605 20$:
05E0 1606 : If we are running in a cluster, ensure that the address specified
05E0 1607 : in the SYSGEN parameter SCSSYSTEMID matches the DECnet node
05E0 1608 : address with the area number. If SCSSYSTEMID contains no area, and
05E0 1609 : the node address is in area 1, don't compare area...
05E0 1610
05E0 1611 TSTL G^CLUSGL_CLUB ; Are we in a cluster?
05E6 1612 BEQL 50$ ; If EQL, no, skip the test
05E8 1613 MOVL R5,R1 ; Make copy of address for CMPW
05EB 1614 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number of SCSSYSTEMID
05ED 1615 #TR4$S_ADDR_AREA,G^SCS$GB_SYSTEMID,R0
05F4 1616 BNEQ 30$ ; If NEQ, compare full address
05F6 1617 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number of node address
05F8 1618 #TR4$S_ADDR_AREA,R5,R0
05FB 1619 CMPL R0,#1 ; Is it area 1?
05FE 1620 BNEQ 40$ ; If NEQ, no: areas do not match.
0600 1621
0600 1622 : Just compare the node address portions.
0600 1623
0600 1624 EXTZV #TR4$V_ADDR_DEST,- ; Get the node within area
0602 1625 #TR4$S_ADDR_DEST,R5,R1
0605 1626
0605 1627 : Note: R1 contains node address, or node with area if area = 1
0605 1628
0605 1629 30$: CMPW R1,G^SCS$GB_SYSTEMID ; Does the node address match?
060C 1630 BEQL 50$ ; If EQL, yes
060E 1631 40$: BRW 190$ ; No, report error
0611 1632 50$:
0611 1633
0611 1634 : If we are acting as a Phase IV router, then if no area number
: has been specified as the executor address, then default it to
```

03 50 E8 05C5 1597 \$GETFLD lni,l,add ; Get the node address
00E8 31 05C8 1598 BLBS R0,5\$; If LBC then report bad node address
55 58 D0 05CB 1599 5\$: MOVL R8,R5 ; Has the address been set yet?
0D 13 05CE 1600 BEQL 10\$; If not set, skip address checks
54 00000000'EF D0 05D0 1601 MOVL NET\$GL_PTR_VCB,R4 ; Get RCB address
OE A4 58 B1 05D7 1602 CMPW R8,RCB\$W_ADDR(R4) ; Is the address being changed?
03 12 05DB 1603 BNEQ 20\$; If so, perform address checks
0107 31 05DD 1604 10\$: BRW 140\$; Skip all address checks
05E0 1605 20\$:
05E0 1606 : If we are running in a cluster, ensure that the address specified
05E0 1607 : in the SYSGEN parameter SCSSYSTEMID matches the DECnet node
05E0 1608 : address with the area number. If SCSSYSTEMID contains no area, and
05E0 1609 : the node address is in area 1, don't compare area...
05E0 1610
05E0 1611 TSTL G^CLUSGL_CLUB ; Are we in a cluster?
05E6 1612 BEQL 50\$; If EQL, no, skip the test
51 55 D0 05E8 1613 MOVL R5,R1 ; Make copy of address for CMPW
0A EF 05EB 1614 EXTZV #TR4\$V_ADDR_AREA,- ; Get the area number of SCSSYSTEMID
50 00000000'GF 06 05ED 1615 #TR4\$S_ADDR_AREA,G^SCS\$GB_SYSTEMID,R0
0F 12 05F4 1616 BNEQ 30\$; If NEQ, compare full address
0A EF 05F6 1617 EXTZV #TR4\$V_ADDR_AREA,- ; Get the area number of node address
50 55 06 05F8 1618 #TR4\$S_ADDR_AREA,R5,R0
01 50 D1 05FB 1619 CMPL R0,#1 ; Is it area 1?
OE 12 05FE 1620 BNEQ 40\$; If NEQ, no: areas do not match.
0600 1621
0600 1622 : Just compare the node address portions.
0600 1623
51 55 00 EF 0600 1624 EXTZV #TR4\$V_ADDR_DEST,- ; Get the node within area
0A 0602 1625 #TR4\$S_ADDR_DEST,R5,R1
0605 1626
0605 1627 : Note: R1 contains node address, or node with area if area = 1
0605 1628
00000000'GF 51 B1 0605 1629 30\$: CMPW R1,G^SCS\$GB_SYSTEMID ; Does the node address match?
03 13 060C 1630 BEQL 50\$; If EQL, yes
0104 31 060E 1631 40\$: BRW 190\$; No, report error
0611 1632 50\$:
0611 1633
0611 1634 : If we are acting as a Phase IV router, then if no area number
: has been specified as the executor address, then default it to

```
0611 1635 ; '1', and set a flag indicating that we should suppress the area
0611 1636 ; number on all node addresses returned to higher layers (NML/EVL).
0611 1637 ;
0611 1638 $GETFLD lni,l,ety ; Get executor type
061E 1639 BLBC R0,60$ ; If not set, then error
03 58 91 0621 1640 CMPB R8,#ADJ$C_PTY_AREA ; Are we an area router?
34 13 0624 1641 BEQL 70$ ; If so, bypass checks
00 55 0A ED 0626 1642 CMPZV #TR4$V_ADDR_AREA,- ; Is the area number 0?
55 06 0628 1643 #TR4$S_ADDR_AREA,R5,#0
3D 12 062B 1644 BNEQ 80$ ; If not, then it's ok
0A 01 FO 062D 1645 INSV #1,#TR4$V_ADDR_AREA,- ; Default it to '1'
55 06 0630 1646 #TR4$S_ADDR_AREA,R5
58 55 DO 0632 1647 MOVL R5,R8 ; Set new address
25 50 E9 0635 1648 $PUTFLD lni,l,add ; Store it
58 01 DO 0642 1649 BLBC R0,80$ ; If error trying to store, skip it
58 50 E9 0645 1650 MOVL #1,R8 ; Remember to suppress area from now on
10 11 0648 1651 $PUTFLD lni,v,sup ; Store the area suppression flag
58 50 E9 0655 1652 60$: BLBC R0,110$ ; Exit if error trying to store
10 11 0658 1653 BRB 80$
065A 1654 ;
065A 1655 ; For area routers, all we need is to make sure the area
065A 1656 ; number is never 0, since it would wreak havoc on the routing
065A 1657 ; algorithms.
065A 1658 ;
065A 1659 70$: $CNFFLD lni,l,add,R9 ; Assume problem with address
52 55 0A EF 0661 1660 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number
55 06 0663 1661 #TR4$S_ADDR_AREA,R5,R2
48 13 0666 1662 BEQL 110$ ; If 0 at this point, then error
10 11 0668 1663 ; (area number can NEVER be zero)
066A 1664 BRB 90$ ; Apply area routing defaults
066A 1665 80$: ;
066A 1666 ; If we are dealing with an area router, or a router which
066A 1667 ; is using explicit area specifications, then provide the
066A 1668 ; additional LNI defaults for area routers.
066A 1669 ;
066A 1670 $GETFLD lni,v,sup ; Is the area suppression flag set?
56 00000000'EF 0A 58 E8 0677 1671 BLBS R8,100$ ; If so, then skip area defaults
FEEF 30 9E 067A 1672 90$: MOVAB NET$G_LNI_AREA,R6 ; Set address of area defaults
0681 1673 BSBW NET$TABLE_DFLT ; Apply the default values
0684 1674 100$: ;
0684 1675 ; Breakup the new executor address into area and node number
0684 1676 ;
52 55 0A EF 0684 1677 EXTZV #TR4$V_ADDR_AREA,- ; Get the area number
55 06 0686 1678 #TR4$S_ADDR_AREA,R5,R2
53 55 00 EF 0689 1679 EXTZV #TR4$V_ADDR_DEST,- ; Get the node within area
0A 068B 1680 #TR4$S_ADDR_DEST,R5,R3
068E 1681 ;
068E 1682 ; Make sure that the area number specified is within MAX AREAS
068E 1683 ;
068E 1684 $GETFLD lni,v,sup ; Is the area suppression flag set?
1A 58 E8 069B 1685 BLBS R8,120$ ; If so, skip check
05 50 E9 069E 1686 $GETFLD lni,l,mar ; Get MAX AREAS
58 52 91 06AB 1687 BLBC R0,110$ ; Branch if not set
05 1B 06AE 1688 CMPB R2,R8 ; Within MAX AREAS?
50 00' 3C 06B1 1689 BLEQU 120$ ; Branch if ok
56 11 06B3 1690 110$: MOVZWL S^#SS$_BADPARAM,R0 ; Indicate error
06B6 1691 BRB 170$ ; Report the error
```



```
06B8 1692 120$: ;
06B8 1693 ; Make sure that the node number specified is within MAX ADDRESS
06B8 1694 ;
06B8 1695 ; GETFLD lni,l, mad ; Get MAX ADDRESS
EB 50 E9 06C5 1696 BLBC R0,110$ ; Branch if not set
58 53 B1 06C8 1697 CMPW R3,R8 ; Within MAX ADDRESS?
E6 1A 06CB 1698 BGTRU 110$ ; Branch if out of range
06CD 1699 ;
06CD 1700 ; Make sure that there are no NDI entries with the new LNI node
06CD 1701 ; address. Refer to NET$INSERT_NDI for more information.
06CD 1702 ;
7E 5A 7D 06CD 1703 MOVQ R10,-(SP) ; Save registers
58 55 D0 06D0 1704 MOVL R5,R8 ; New executor address
OE4F 30 06D3 1705 BSBW NET$NDI_BY_ADD ; Lookup NDI for this address
00000000'EF 5A D1 06D6 1706 CMPL R10,NET$GL_LOCAL_NDI ; Did we find the local NDI?
02 12 06DD 1707 BNEQ 130$ ; If so, branch
50 D4 06DF 1708 CLRL R0 ; Else, indicate address already in use
5A 8E 7D 06E1 1709 130$: MOVQ (SP)+,R10 ; Restore LNI pointers
28 50 E8 06E4 1710 BLBS R0,180$ ; Error if already in use
06E7 1711 140$: ;
06E7 1712 ; Update the ACP control layer
06E7 1713 ;
06E7 1714 ; Inputs: R11 LNI CNR pointer
06E7 1715 ; R10 New LNI CNF pointer
06E7 1716 ;
06E7 1717 ; Outputs: R9 I.D. of faulty parameter if LBC in R0
06E7 1718 ; R0 Status
06E7 1719 ;
06E7 1720 ; All other regs may be clobbered.
06E7 1721 ;
06E7 1722 ;
F916' 30 06E7 1723 150$: BSBW NET$UPD_LOCAL ; Make ACP transistion
21 50 E9 06EA 1724 BLBC R0,170$ ; If error detected, then exit
OC 58 E9 06ED 1725 $GETFLD lni,v,sup ; If areas were suppressed,
54 00000000'EF D0 06FA 1726 BLBC R8,160$ ;
008C C4 01 90 06FD 1727 MOVL NET$GL_PTR_VCB,R4 ; Get RCB address
0704 1728 MOVB #1,RCB$B_MAX_AREA(R4) ; Stuff Max Area to 1, but not in the
0709 1729 160$: ;
0709 1730 ; Now that there is no more possibility for error, we must
0709 1731 ; reposition the fake 'area.0' NDI CNF which marks the position
0709 1732 ; in the NDI linked list of the beginning of the current area.
0709 1733 ; (This marker is needed to determine when a scan needs to shift
0709 1734 ; to using the OA vector or not). If no marker currently
0709 1735 ; exists, then one is created assuming that this is the first
0709 1736 ; LNI insertion.
0709 1737 ;
50 12 10 0709 1738 BSBB NDI_MARKER ; Insert the marker in the right place
50 00' D0 070B 1739 MOVL S^#SS$ _NORMAL,R0 ; CNF, where it might be seen by user
05 070E 1740 170$: RSB
070F 1741 ;
50 0000'8F 3C 070F 1742 180$: MOVZWL #SS$ _DEACTIVE,R0 ; Executor address is used elsewhere
05 0714 1743 RSB
0715 1744 ;
50 00000000'8F D0 0715 1745 190$: MOVL #SS$ _WRONGNAME,R0 ; Executor address does not match
05 071C 1746 RSB ; VAXcluster system ID.
```

```
071D 1748 .SBTTL NDI_MARKER - Insert executor NDI marker
071D 1749 :+
071D 1750 : NDI_MARKER - Insert marker into NDI linked list for executor node
071D 1751 :
071D 1752 : This routine inserts a dummy CNF block into the NDI linked list at
071D 1753 : the position where the executor address would normally go. This marker
071D 1754 : is needed when scanning the database in order to know when to begin
071D 1755 : using the OA vector, rather than the linked list, and where to
071D 1756 : return again after finishing with the vector. See SCAN_NDI for more
071D 1757 : details.
071D 1758 :
071D 1759 : Inputs:
071D 1760 :
071D 1761 : RCB$W_ADDR = New executor address
071D 1762 :
071D 1763 : Outputs:
071D 1764 :
071D 1765 : None
071D 1766 :
071D 1767 : All registers are preserved.
071D 1768 :-
071D 1769 NDI_MARKER:
071D 1770 PUSHRR #M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Save registers
5B 00000000'EF BB 0721 1771 MOVL NET$GL_CNR_NDI,R11 ; Point to NDI root
    SA 5B DO 0728 1772 MOVL R11,R10 ; Start at beginning of tree
    072B 1773 :
    072B 1774 : The executor node address is always stored as 0, in the
    072B 1775 : collating tree. Therefore we will insert a marker in the
    072B 1776 : collating tree as n.0 where n is the current area field
    072B 1777 : of the executor node address. To make finding this NDI as
    072B 1778 : easy as possible, it will be searched in the name tree as
    072B 1779 : "+++". Since this is an invalid node name string, it will
    072B 1780 : be a unique entry in the name tree.
    072B 1781 :
    7E 2B2B2B2B 8F DO 072B 1782 MOVL #A'++++',-(SP) ; Store string on the stack
        57 04 9A 0732 1783 MOVZBL #4,R7 ; Set string length
        58 5E DO 0735 1784 MOVL SP,R8 ; Point R8 to string
        F8C5' 30 0738 1785 BSBW NET$FIND_NAME ; Search the name tree for NDI
        5E 04 CO 073B 1786 ADDL #4,SP ; Cleanup stack
        05 50 E9 073E 1787 BLBC R0,20$ ; Br if not found
        F8BC' 30 0741 1788 BSBW NET$DELETE_BTE ; Else, delete the old BTEs
        21 11 BRB 40$ ; And continue
    0746 1790 :
    0746 1791 : If the old NDI cannot be found, then we will create one.
    0746 1792 :
    0746 1793 20$: MOVZWL CNR$W_SIZ_CNF(R11),R1 ; Set size of CNF block
    074A 1794 ADDL #8,R1 ; Add in enough room for name
    00000000'EF 16 074D 1795 JSB NET$ALLOCATE ; Allocate block from ACP pool
        52 50 E9 0753 1796 BLBC R0,90$ ; Br if error, skip it
        SA 52 DO 0756 1797 MOVL R2,R10 ; Save address of marker CNF
        F8A4' 30 0759 1798 BSBW CNF$INIT ; Initialize CNF block
        40 8F 88 075C 1799 BISB #1&NDI_V_MARKER,- ; Initialize flags
        08 AA 075F 1800 CNF$B_FLG(R10)
        18 AA 7C 0761 1801 CLRQ CNF$B_MASK(R10) ; Initialize 12 byte bitmask
        20 AA D4 0764 1802 CLRL CNF$B_MASK+8(R10)
    0767 1803 40$: :
    0767 1804 : Common processing
```

55	00000000	'EF	D0	0767	1805	:		
	58	0E A5	3C	0767	1806	MOVL	NET\$GL_PTR_VCB,R5	; Get RCB address
	12	AA 58	B0	076E	1807	MOVZWL	RCB\$W_ADDR(R5),R8	; Get executor node address
		00 00	F0	0772	1808	MOVW	R8,CNF\$W_ID(R10)	; Copy/reset new executor address
		58 0A		0776	1809	INSV	#0,#TR4\$V_ADDR_DEST,-	; Zero the address within the area
				0779	1810		#TR4\$S_ADDR_DEST,R8	
				077B	1811	\$PUTFLD	ndi,l,add	; Store it in CNF
7E	2B2B2B2B	8F	D0	0788	1812	MOVL	#^A'++++',-(SP)	; Store string on the stack
	57	04	9A	078F	1813	MOVZBL	#4,R7	; Set string length
	58	5E	D0	0792	1814	MOVL	SP,R8	; Point R8 to string
				0795	1815	\$PUTFLD	ndi,s,nna	; Set the node name field
	5E	04	C0	07A2	1816	ADDL	#4,SP	; Cleanup stack
	F858		30	07A5	1817	BSBW	NEI\$ADD_NDI	; Add new NDI to trees
	0FFF	8F	BA	07A8	1818	POPR	#^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>	; Restore registers
			05	07AC	1819	RSB		

```
07AD 1821 .SBTTL NET$INSERT_NDI - PRE-INSERTION PROCESSING
07AD 1822 :+
07AD 1823 : NET$INSERT_NDI - Insert an NDI entry into database
07AD 1824 :
07AD 1825 : This routine is called to validate the CNF entry before inserting
07AD 1826 : it into the database.
07AD 1827 :
07AD 1828 : Inputs:
07AD 1829 :
07AD 1830 : R11 = Address of CNR
07AD 1831 : R10 = Address of CNF
07AD 1832 :
07AD 1833 : Outputs:
07AD 1834 :
07AD 1835 : R0 = Status code. If error, entry is not inserted.
07AD 1836 :-
07AD 1837 .ENABL LSB
07AD 1838
07AD 1839 NET$INSERT_NDI::
07AD 1840 :
07AD 1841 : Use this opportunity to keep the NDI and LNI databases consistent
07AD 1842 : by using the following rules. These rules are needed because the
07AD 1843 : the local node address exists in both data bases. By forcing the
07AD 1844 : address of the local node NDI and all loop nodes to be zero and by
07AD 1845 : using a zero NDI node address to implicitly refer to the RCB$W_ADDR
07AD 1846 : value, changing the local node address in the LNI data base will
07AD 1847 : automatically update the addresses of the peritinent NDI entries.
07AD 1848 : The fact that the local node's address is stored as zero throughout
07AD 1849 : the NDI data base is used throughout the ACP -- it will be very
07AD 1850 : difficult to modify the design.
07AD 1851 :
07AD 1852 : Upon new NDI insertion:
07AD 1853 :
07AD 1854 : 1. if new NDI node address = RCB$W_ADDR
07AD 1855 : then zero the NDI node address
07AD 1856 :
07AD 1857 : 2. if old NDI node address = 0
07AD 1858 : then
07AD 1859 : if new NDI node address NEQ 0
07AD 1860 : then error
07AD 1861 :
07AD 1862 : Upon new LNI insertion: (see UPD_LOCAL above)
07AD 1863 :
07AD 1864 : 1. if new LNI node address exists anywhere in the
07AD 1865 : NDI data base
07AD 1866 : then error
07AD 1867 :
07AD 1868 : MOVL R10,R5 ; Save the new NDI
07AD 1869 : BICB #<1@NDI_V_LOCAL>!-- ; Init special flags
07AD 1870 : <1@NDI_V_LOOP>!--
07AD 1871 : CNF$B F[GR5)
07AD 1872 : BSBW CHK_LOGIN_NDI ; Check default login strings for
07AD 1873 : ; combined length
07AD 1874 : BLBC R0,5$ ; If LBC then too long
07AD 1875 : MOVL R4,NDI_L_NACS ; Remember number of non-null access
07AD 1876 : ; control strings
07AD 1877 : $GETFLD ndi,s,nna ; Get the node name descriptor
```

55 5A DO 07AD 1868
8A 07AD 1869
07AD 1870
07AD 1871
07AD 1872
07AD 1873
07AD 1874
07AD 1875
07AD 1876
07AD 1877

OB A5 30 07AD 1871
0264 30 07AD 1872
7A 50 E9 07AD 1874
00000000'EF 54 DO 07AD 1875

```
00000018'EF 57 7D 07CE 1878      MOVQ    R7,NDI_Q_NAME      ; Save it
                                07D5 1879      $GETFLD ndi,l,add      ; Get new node address
                                58 D5 07E2 1880      TSTL    R8          ; Address = 0?
                                51 13 07E4 1881      BEQL    10$          ; If so, then br (for loop nodes)
52 00000000'EF D0 07E6 1882      MOVL    NET$GL_PTR_VCB,R2      ; Get the RCB pointer
                                00 EF 07ED 1883      EXTZV    #TR4$V-ADDR_DEST,- ; Get the node # within area
                                50 58 0A 07EF 1884      EXTZV    #TR4$S-ADDR_DEST,R8,R0
                                51 58 0A EF 07F2 1885      EXTZV    #TR4$V-ADDR_AREA,- ; Get the area number
                                14 12 07F4 1886      EXTZV    #TR4$S-ADDR_AREA,R8,R1
                                008B C2 F0 07F7 1887      BNEQ    4$          ; Check it if non-zero
                                58 0A 07FD 1888      INSV     RCB$B_HOMEAREA(R2),- ; If area = 0, then use our area
                                F7FD' 30 07FE 1889      INSV     #TR4$V-ADDR_AREA,- ; (accept 0 as synonym for our area)
                                00 EF 0800 1891      BSBW     CNF$PUT_FIELD      ; Replace node address
                                50 58 0A 0803 1892      EXTZV    #TR4$V-ADDR_DEST,- ; Restore the node # within area
                                51 008B C2 9A 0805 1893      EXTZV    #TR4$S-ADDR_DEST,R8,R0
                                008C C2 51 91 0808 1894      MOVZBL  RCB$B_HOMEAREA(R2),R1 ; Use our area as the area number
                                5B 1A 080D 1895      CMPB     R1,RCB$B_MAX_AREA(R2) ; Check against max allowed area
                                008B C2 51 91 0812 1896      BGTRU    30$          ; If GTRU then out of range
                                5A A2 50 B1 0814 1897      CMPB     R1,RCB$B_HOMEAREA(R2) ; Is this our area?
                                0E A2 58 B1 0819 1898      BNEQ    10$          ; If not, then no limit on max address
                                1C 12 081B 1899      CMPW     R0,RCB$W_MAX_ADDR(R2) ; Check against max allowed address
                                4E 1A 081F 1900      BGTRU    30$          ; If GTRU then out of range
                                0E A2 58 B1 0821 1901      CMPW     R8,RCB$W_ADDR(R2) ; Is it the local node?
                                1C 12 0825 1902      BNEQ    10$          ; If NEQ no
                                58 D4 0827 1903      CLRL     R8          ; Local node is stored in the NDI data
                                F7D4' 30 0829 1904      BSBW     CNF$PUT_FIELD      ; base as zero
                                08 50 E8 082C 1905      BLBS     R0,10$          ; If LBS, continue
50 0000'8F 3C 082F 1906      MOVZWL  #SS$-INSFMEM,R0 ; Report "insufficient memory"
                                0189 31 0834 1907      BRW     220$          ; Take common exit
                                12 A5 58 B0 0837 1908      MOVW     R8,CNF$W_ID(R5) ; Save the new NDI node address
                                083B 1909      ;
                                083B 1910      ;
                                083B 1911      ;
                                083B 1912      ;
                                083B 1913      ;
                                083B 1914      ;
                                083B 1915      ;
                                083B 1916      ;
                                083B 1917      ;
                                083B 1918      ;
                                083B 1919      ;
                                083B 1920      ;
                                083B 1921      ;
                                083B 1922      ;
                                083B 1923      ;
                                083B 1924      ;
                                083B 1925      ;
                                083B 1926      ;
                                083B 1927      ;
                                083B 1928      ;
                                083B 1929      ;
                                083B 1930      ;
                                083B 1931      ;
                                083B 1932      ;
                                083B 1933      ;
                                083B 1934      ;
```

A "loop" node is an NDI for which an output line has been permanently assigned in the database. Such an NDI must have a name field (ndi,s,nna) specified and, for now, must have the node address of the local node. By creating a logical link to this nodename, the link is made to the local node but all traffic is transmitted over the specified line. The intent of "loop" nodes is to allow loopback testing of a line or the testing of the transport layer on the node at the other end of the line.

A "loop" node NDI cannot be converted to a normal NDI or vice-versa. The rules governing NDI updates with respect to the associated loopback linename are as follows:

```
      If there's a loopback line associated with the new NDI
      then
        if the old NDI is a "loop" node
        then
          if new NDI node address = 0
          then okay
          else node address is invalid
        else
          loopback line is an invalid parameter
      else
        if the old NDI was a "loop" node
```

```

                                then
                                mark the new NDI for delete and return success
                                else
                                neither old nor new NDI are "loop" nodes, continue

                                083B 1935
                                083B 1936
                                083B 1937
                                083B 1938
                                083B 1939
                                083B 1940
                                083B 1941
                                32 50 E9 0848 1942 $GETFLD ndi,s,nli ; Get optional circuit name
                                12 AA B5 084B 1943 BLBC R0,NOT_LOOPNODE ; If LBC, new NDI not a "loop"
                                2D 12 084E 1944 TSTW CNFSW_ID(R10) ; Non-zero node address?
                                10 88 0850 1945 BNEQ NOT_LOOPNODE ; Loop nodes always use address 0
                                0B AA 0852 1946 BISB #1@NDI_V_LOOP,- ; Mark it as being a "loop" node
                                56 D5 0854 1947 CNFSB_FLG(R10)
                                05 13 0856 1948 TSTL R6 ; Is there an old NDI ?
                                04 E1 0858 1949 BEQL 20$ ; If EQL no
                                12 0B A6 085A 1950 BBC #NDI_V_LOOP,- ; If BC, old was not a loop node
                                00000018'EF D5 085D 1951 20$: $CNFFLD ndi,s,nna,R9 ; - therefore cannot set linename
                                09 13 086A 1952 TSTL NDI_Q_NAME ; Assume no name was specified
                                014E 31 086C 1953 BEQL 35$ ; Is the name null?
                                086F 1954 BRW 210$ ; If EQL report 'insufficient args'
                                50 00' 3C 086F 1955 30$: MOVZWL S^#SS$ _BADPARAM,R0 ; Indicate error
                                014B 31 0872 1957 BRW 220$ ; Take common exit
                                50 0000'8F 3C 0875 1958 35$: MOVZWL #SS$ _INSFARG,R0 ; Set error code
                                0143 31 087A 1959 BRW 220$ ; Take common exit
                                087D 1960
                                56 D5 087D 1961 NOT_LOOPNODE:
                                0C 13 087F 1962 TSTL R6 ; Is there an old NDI ?
                                04 E1 0881 1963 BEQL 110$ ; If EQL no
                                07 0B A6 0883 1964 BBC #NDI_V_LOOP,- ; If BC old was not a loop node
                                88 0886 1965 BISB #CNFSM_FLG_DELETE!- ; Mark new NDI for delete
                                08 AA 12 0887 1967 <1@NDI_V_LOOP>,- ; ...it's still a "loop node"
                                0130 31 088A 1968 CNFSB_FLG(R10)
                                088D 1970 BRW 210$ ; Report success
                                088D 1971 110$:
                                088D 1972 ; Neither the old or new NDIs are loop nodes
                                12 A5 B5 088D 1973 TSTW CNFSW_ID(R5) ; Is this the local node?
                                04 12 0890 1974 BNEQ 120$ ; If NEQ then no
                                20 88 0892 1975 BISB #1@NDI_V_LOCAL,- ; Mark it as "local"
                                0B A5 0894 1976 CNFSB_FLG(R5)
                                56 D5 0896 1977 120$: TSTL R6 ; Is there an old NDI
                                14 13 0898 1978 BEQL 130$ ; If EQL then no
                                12 A6 12 A5 B1 089A 1979 CMPW CNFSW_ID(R5),CNFSW_ID(R6) ; Are old and new address the same?
                                0D 13 089F 1980 BEQL 130$ ; If so, branch
                                08A1 1981
                                08A1 1982 ; The old and new address are different. If either has address "zero"
                                08A1 1983 ; then report that the local NDI is being falsely modified.
                                08A1 1984
                                05 E0 08A1 1985 BBS #NDI_V_LOCAL,- ; If BS then old NDI is local
                                1E 0B A6 08A3 1986 -CNFSB_FLG(R6),140$ ; -- report error
                                05 E0 08A6 1987 BBS #NDI_V_LOCAL,- ; If BS then new NDI is local
                                19 0B A5 08A8 1988 -CNFSB_FLG(R5),140$ ; -- report error
                                009D 31 08AB 1989 125$: BRW 200$ ; Insert new NDI into vector
                                08AE 1990 130$:
                                08AE 1991 ; There has been no NDI node address change.
```

```
08AE 1992
08AE 1993
08AE 1994
08AE 1995
08AE 1996
08AE 1997
08AE 1998
08AE 1999
08AE 2000
08AE 2001
08AE 2002
08AE 2003
08AE 2004
08AE 2005
08AE 2006
08AE 2007
08AE 2008
08AE 2009
08AE 2010
08AE 2011
05 E1 08AE 2012
F8 0B A5 08B0 2013
00000000'EF D5 08B3 2014
F0 13 08B9 2015
56 00000000'EF D1 08BB 2016
0F 13 08C2 2017
50 0000'8F 3C 08C4 2018 140$:
00ED 31 08CB 2019
08D0 2020
08D3 2021 145$:
08D3 2022
08D3 2023
08D3 2024
08D3 2025
08D3 2026
08D3 2027
52 00000000'EF D0 08D3 2028 150$:
OE A2 B5 08DA 2029
08DD 2030
6C 13 08DD 2031
57 00000018'EF 7D 08DF 2032
08E6 2033
08E6 2034
08E6 2035
08E6 2036
00000000'GF D5 08E6 2037
16 13 08EC 2038
20 68 57 2D 08EE 2039
00000000'GF
0A 13 08F8 2040
50 00000000'8F D0 08FA 2041
00BC 31 0901 2042
0904 2043
0904 2044
0904 2045
51 00000028'EF 9E 0904 2046 160$:
00000024'EF 51 D0 090B 2047

: If the current local NDI CNF address = NET$GL_DUM_NDI
: (which means the ACP is undergoing its initialization cycle)
:
: OR
:
: the new NDI node address not= 0 (0 implies NDI for local node)
: then
: we're done, return success
:
: else
: if 'old' NDI CNF address = current local NDI CNF address
: then
: the local node NDI is undergoing a valid update.
: else
: the local node NDI is being falsely modified, i.e., the
: NDI specified via the Q10 P2 key was not the local node
: NDI but the node address parameter specified in the Q10 P4
: equals the local node address. Report the error.
:
BBC #NDI_V_LOCAL,- : If BC then new NDI is not for the
CNF$B_FLG(R5),125$ : local node -- we're done
TSTL NET$GL_LOCAL_NDI : Has the local NDI been init'd yet?
BEQL 125$ : If EQL no, we're done
CMPL NET$GL_LOCAL_NDI,R6 : Is the 'old' CNF really the current
: local node CNF
BEQL 150$ : If EQL yes, br to continue
$CNFFLD ndi,l,add,r9 : Specify field i.d. as node address
MOVZWL #SS$_DEACTIVE,R0 : Specify it as already in use
: elsewhere in the database
: Report error
BRW 220$
:
: Update the logical name for the local node only if the RCB contains
: a non-zero node address. The RCB contains node address 0 while the
: ACP is initializing.
:
MOVL NET$GL_PTR_VCB,R2 : Get the RCB address
TSTW RCB$_ADDR(R2) : Has the local node address been
: established yet?
BEQL 200$ : If EQL no, we're done
MOVQ NDI_Q_NAME,R7 : Get NDI name descriptor
:
: If this is a VAXcluster, the node name must match the name set up
: in the SYSGEN parameter SCSNODE.
:
TSTL G^CLUSGL_CLUB : See if this is a VAXcluster
BEQL 160$ : If EQL, no - don't check name
CMPCS R7,(R8),#^A' ',#6,G^SCS$GB_NODENAME : Verify that names match
:
BEQL 160$ : If EQL, names match
MOVL #SS$_WRONGNAME,R0 : Else, report error
BRW 220$ : ...
:
: Create SYSSNODE logical name
:
MOVAB NDI_LNAMEBUF,R1 : Get buffer for building logical name
MOVL R1,NDI_Q_LNAME+4 : Setup pointer in descriptor
```

```
00000020'EF 51 CE 0912 2048 MNEGL R1,NDI_Q_LNAME ; Bias the count field
      81 5F 8F 90 0919 2049 MOVVB #^A'^(R1)+ ; Build logical nodename
      81 03 11 091D 2050 BRB 180$- ; Continue
      FA 57 90 091F 2051 170$: MOVVB (R8)+,(R1)+ ; Enter the name text
      81 3A3A 8F 80 0922 2052 180$: SOBGEQ R7,170$ ; Loop
00000020'EF 51 C0 0925 2053 MOVW #^A'::^(R1)+ ; Enter nodename delimiter
      092A 2054 ADDL R1,NDI_Q_LNAME ; Setup the count field
      0931 2055 $CRELOG_S ; Create a system logical name
      0931 2056 LOGNAM = SYSNODE DESC,- ; Logical name
      0931 2057 EQLNAM = NDI_Q_LNAME ; Equivalence name
      75 50 E9 0948 2058 BLBC R0,220$ ; If LBC then error
      094B 2059 ;
      094B 2060 ; Update the logical name for the alias node only if the RCB contains
      094B 2061 ; a non-zero alias address.
      094B 2062 ;
52 00000000'EF D0 094B 2063 200$: MOVL NET$GL_PTR_VCB,R2 ; Get the RCB address
      008D C2 B5 0952 2064 TSTW RCB$W_ALIAS(R2) ; Is there an alias local address?
      008D C2 12 A5 B1 0956 2065 BEQL 204$ ; If not, we're done
      4E 12 095E 2066 CMPW CNF$W_ID(R5),RCB$W_ALIAS(R2) ; Is this the alias NDI entry?
      57 00000018'EF 7D 0960 2067 BNEQ 204$ ; Skip if not
51 00000028'EF 9E 0967 2068 MOVQ NDI_Q_NAME,R7 ; Get NDI name descriptor
00000024'EF 51 D0 096E 2069 MOVAB NDI_LNAMEBUF,R1 ; Get buffer for building logical name
00000020'EF 51 CE 0975 2070 MOVL R1,NDI_Q_LNAME+4 ; Setup pointer in descriptor
      81 5F 8F 90 097C 2071 MNEGL R1,NDI_Q_LNAME ; Bias the count field
      03 11 0980 2072 MOVVB #^A'^(R1)+ ; Build logical nodename
      81 88 90 0982 2073 BRB 203$- ; Continue
      FA 57 F4 0985 2074 202$: MOVVB (R8)+,(R1)+ ; Enter the name text
      81 3A3A 8F 80 0988 2075 203$: SOBGEQ R7,202$ ; Loop
00000020'EF 51 C0 098D 2076 MOVW #^A'::^(R1)+ ; Enter nodename delimiter
      0994 2077 ADDL R1,NDI_Q_LNAME ; Setup the count field
      0994 2078 $CRELOG_S ; Create a system logical name
      0994 2079 LOGNAM = CLUNODE DESC,- ; Logical name
      12 50 E9 0994 2080 EQLNAM = NDI_Q_LNAME ; Equivalence name
      09AB 2081 BLBC R0,220$ ; If LBC then error
      09AE 2082 204$: ;
      09AE 2083 ; New will be inserted into the data base, start/reset counter timer.
      09AE 2084 ;
      12 A5 B5 09AE 2085 TSTW CNF$W_ID(R5) ; Is this the local node's NDI?
      07 12 09B1 2086 BNEQ 205$ ; If so,
00000000'GF 55 D0 09B3 2087 MOVL R5,G^NET$GL_LOCAL_NDI ; Remember address of local NDI CNF
      F643' 30 09BA 2088 205$: BSBW NET$SET_CTR_TIMER ; Start/reset automatic counter timer
      50 01 D0 09BD 2089 210$: MOVL #1,R0 ; Success if not changing local node
      05 09C0 2090 220$: RSB ; Done
      09C1 2091 ;
      09C1 2092 .DSABL LSB
```



```
09C1 2094 .SBTTL NET$INSERT_OBI - PRE-INSERTION PROCESSING
09C1 2095 :+
09C1 2096 : NET$INSERT_OBI - Insert an OBI entry into database
09C1 2097 :
09C1 2098 : This routine is called to validate the CNF entry before inserting
09C1 2099 : it into the database.
09C1 2100 :
09C1 2101 : Inputs:
09C1 2102 :
09C1 2103 : R11 = Address of CNR
09C1 2104 : R10 = Address of CNF
09C1 2105 :
09C1 2106 : Outputs:
09C1 2107 :
09C1 2108 : R0 = Status code. If error, entry is not inserted.
09C1 2109 :-
09C1 2110
004C 30 09C1 2111 NET$INSERT_OBI:: ; New CNF OBI special processing
23 50 E9 09C1 2112 BSBW CHK_LOGIN_OBI ; Check default login strings for
09C4 2113 ; combined length
09C4 2114 BLBC R0,20$ ; If LBC then too long
09C7 2115 :
09C7 2116 : If an OBI is merely the result of a 'declare name/object' QIO then
09C7 2117 : when the channel over which the object is declared is broken it is
09C7 2118 : appropriate to delete the OBI entry from the data base. However,
09C7 2119 : in order to prevent a defined (via NCP) object from being removed
09C7 2120 : if it is subsequently declared and then 'undeclared' (by having its
09C7 2121 : associated channel broken) it is necessary to mark each OBI if it
09C7 2122 : at any time exists without being in a 'declared' state.
09C7 2123 :
09C7 2124 : Note that it is sufficient to mark the OBI whenever it is being
09C7 2125 : (re)inserted into the database and it is not currently declared.
09C7 2126 :
09C7 2127 $GETFLD obi,l,ucb ; Is the OBI currently 'declared'
10 50 E8 09D4 2128 BLBS R0,10$ ; If LBS yes, don't mark it
58 01 D0 09D7 2129 MOVL #1,R8 ; Set next bit value to 'true'
09DA 2130 $PUTFLD obi,v,set ; Mark the OBI as having existed
09E7 2131 ; without being declared
50 01 D0 09E7 2132 10$: MOVL #1,R0 ; Set success
05 09EA 2133 20$: RSB
```

```
09EB 2135 .SBTTL NET$INSERT_xxx - PRE-INSERTION PROCESSING
09EB 2136 ;+
09EB 2137 ; NET$INSERT_xxx - Insert an entry into database
09EB 2138 ;
09EB 2139 ; This routine is called to validate the CNF entry before inserting
09EB 2140 ; it into the database.
09EB 2141 ;
09EB 2142 ; Inputs:
09EB 2143 ;
09EB 2144 ; R11 = Address of CNR
09EB 2145 ; R10 = Address of CNF
09EB 2146 ;
09EB 2147 ; Outputs:
09EB 2148 ;
09EB 2149 ; R0 = Status code. If error, entry is not inserted.
09EB 2150 ; -
09EB 2151
09EB 2152 NET$INSERT_ESI:: ; New ESI CNF special processing
F612' 31 09EB 2153 BRQ NET$DBC_ESI ; Event logger Sink database change
F60F' 31 09EE 2154 NET$INSERT_EFI:: ; New EFI CNF special processing
09EE 2155 BRQ NET$DBC_EFI ; Event logger Filter database change
09F1 2156
09F1 2157 NET$INSERT_LLI:: ; Insert LLI special processing
09F1 2158 NET$INSERT_SPI:: ; Insert SPI database entry
50 01 D0 09F1 2159 MOVL #1,R0 ; No special checking
05 09F4 2160 RSB
09F5 2161
09F5 2162 NET$INSERT_AJI:: ; Insert AJI special processing
50 0000'8F 3C 09F5 2163 MOVZWL #SS$_ILLCNTRFUNC,R0 ; Illegal ACP control function
59 13 9A 09FA 2164 MOVZBL #NFB$C_DB_AJI,R9 ; Qualify error -- not valid for this
05 09FD 2165 ; data base
09FD 2166 RSB ; Done
09FE 2167
09FE 2168 NET$INSERT_SDI:: ; Insert SDI special processing
50 0000'8F 3C 09FE 2169 MOVZWL #SS$_ILLCNTRFUNC,R0 ; Illegal ACP control function
59 1A 9A 0A03 2170 MOVZBL #NFB$C_DB_SDI,R9 ; Qualify error -- not valid for this
05 0A06 2171 ; data base
0A06 2172 RSB ; Done
0A07 2173
0A07 2174 NET$INSERT_ARI:: ; Insert ARI special processing
50 0000'8F 3C 0A07 2175 MOVZWL #SS$_ILLCNTRFUNC,R0 ; Illegal ACP control function
59 14 9A 0A0C 2176 MOVZBL #NFB$C_DB_ARI,R9 ; Qualify error -- not valid for this
05 0A0F 2177 ; data base
0A0F 2178 RSB ; Done
```

```

      0A10 2180      .SBTTL  CHK_LOGIN_xxx - CHECK LOGIN STRING LENGTH
      0A10 2181      :+
      0A10 2182      : CHK_LOGIN_xxx - Check if access control string is too long
      0A10 2183      :
      0A10 2184      : Inputs:
      0A10 2185      :
      0A10 2186      :     R11 = Address of CNR
      0A10 2187      :     R10 = Address of CNF
      0A10 2188      :
      0A10 2189      : Outputs:
      0A10 2190      :
      0A10 2191      :     R0 = Status code
      0A10 2192      : -
      0A10 2193      : .ENABL  LSB
      0A10 2194
      0A10 2195      CHK_LOGIN_OBI:
      0A10 2196      CRL  R4
      53 00000048'EF 9E 0A12 2197      MOVAB  OBI_LOGIN_VEC,R3
      12 11 0A19 2198      BRB  5$
      0A1B 2199      CHK_LOGIN_NDI:
      0A1B 2200      CRL  R4
      53 00000038'EF 9E 0A1D 2201      MOVAB  NDI_PLOGIN_VEC,R3
      OD 10 0A24 2202      BSBB  10$
      53 00000028'EF 9E 0A26 2203      MOVAB  NDI_NLOGIN_VEC,R3
      04 10 0A2D 2204      5$: BSBB  10$
      50 00 0A2F 2205      7$: MOVL  S^#SS$_NORMAL,R0
      05 0A32 2206      RSB
      0A33 2207
      0A33 2208
      52 3C 3C 0A33 2209      10$: MOVZWL #ICB$_ACCESS-4,R2
      0A36 2210
      0A36 2211
      59 83 D0 0A36 2212      20$: MOVL  (R3)+,R9
      14 13 0A39 2213      BEQL  30$
      F5C2' 30 0A3B 2214      BSBW  CNF$GET_FIELD
      F5 50 E9 0A3E 2215      BLBC  R0,20$
      54 D6 0A41 2216      INCL  R4
      52 57 C2 0A43 2217      SUBL  R7,R2
      EE 18 0A46 2218      BGEQ  20$
      50 0000'8F 3C 0A48 2219      MOVZWL #SS$_BADPARAM,R0
      0A4D 2220
      8E D5 0A4D 2221      TSTL  (SP)+
      05 0A4F 2222      30$: RSB
      0A50 2223
      0A50 2224      .DSABL  LSB
      :
      : Check combined login string length
      : Count of number of non-null strings
      : Setup field i.d. vector
      : Continue in common
      : Check combined login string length
      : Count of number of non-null strings
      : Setup field i.d. vector
      : Check the combined length
      : Get the address of field i.d.'s
      : Check the comined length
      : Inicate success
      :
      : Get max size of string text (the -4
      : is for 3 string count fields plus a
      : count field for the combined strings)
      : Get next field i.d.
      : If EQL then done
      : Get the string
      : If LBC then string is not defined
      : Count number of non-null strings
      : Subtract string size
      : If GEQ then okay
      : Else indicate error
      : a better error code is needed
      : Pop stack to return to origin. caller
      : Return
```

```
0A50 2226 .SBTTL NET$SPCINS_XXX - SPECIAL DATABASE INSERTION ROUTINES
0A50 2227 .SBTTL NET$SPCINS_DEF - DEFAULT DATABASE INSERTION ROUTINE
0A50 2228 :+
0A50 2229 : NET$SPCINS_DEF - Default database insertion routine
0A50 2230 :
0A50 2231 : This routine is called to insert a CNF entry in the standard linked list.
0A50 2232 :
0A50 2233 : Inputs:
0A50 2234 :
0A50 2235 :     R11 = Address of CNR
0A50 2236 :     R10 = Address of CNF
0A50 2237 :     R6  = Address of old CNF (or 0 if no old CNF)
0A50 2238 :
0A50 2239 : Outputs:
0A50 2240 :
0A50 2241 :     R0 = Always True.
0A50 2242 :
0A50 2243 :     R5,R9 are destroyed.
0A50 2244 :
0A50 2245 :-
0A50 2246
0A50 2247 NET$SPCINS_CRI:: : Circuit CNF insertion routine
0A50 2248 NET$SPCINS_PLI:: : Line CNF insertion routine
0A50 2249 NET$SPCINS_LNI:: : Local node CNF real insertion routine
0A50 2250 NET$SPCINS_OBI:: : Object CNF insertion routine
0A50 2251 NET$SPCINS_EFI:: : Event filter CNF insertion routine
0A50 2252 NET$SPCINS_ESI:: : Event sink CNF insertion routine
0A50 2253 NET$SPCINS_SPI:: : Server process CNF insertion routine
0A50 2254 NET$SPCINS_LLI:: : Logical link CNF insertion routine
0A50 2255 NET$SPCINS_AJI:: : Adjacency CNF insertion routine
0A50 2256 NET$SPCINS_SDI:: : DLE CNF insertion routine
0A50 2257 NET$SPCINS_ARI:: : AREA Adjacency CNF insertion routine
0A50 2258 NET$SPCINS_DEF:: : Insert block
0A50 2259      MOVL CNR$ FLD COLL(R11),R9 : Get the collating field i.d.
0A54 2260      BSBW CNF$GET_FIELD : Get the field value/descriptor
0A57 2261      MOVL R10,R5 : Save ptr to 'new' CNF
0A5A 2262      CLRL R10 : Start search from head of list
0A5C 2263      MOVL S^#NFB$C_OP_FNDPOS,R1 : Search database to find the CNF after
0A5F 2264      BSBW CNF$KEY_SEARCH : which to insert the new CNF
0A62 2265      BLBS R0,50$ : If LBS then successful
0A65 2266      MOVL CNR$ BLINK(R11),R10 : Else locate last CNF in the queue
0A69 2267 50$: INSQUE (R5),CNF$ FLINK(R10) : Insert after item found
0A6C 2268      MOVL R5,R10 : Point to the 'new' CNF
0A6F 2269      MOVL S^#SS$_NORMAL,R0 : Indicate success
0A72 2270      TSTL R6 : Is there an 'old' CNF ?
0A74 2271      BEQL 70$ : If EQL no
0A76 2272      BBS #CNF$V_FLG_ACP,- : If BS then CNF is not linked into the
0A78 2273      CNF$B_FLG(R6),70$ : CNF queue
0A7B 2274      REMQUE (R6),R5 : Remove 'old' CNF from database
0A7E 2275 60$: INSQUE (R5),@NET$GQ_TMP_BUF : Queue CNF block for deallocation
0A85 2276 70$: RSB : Return to caller
```

59 14 AB D0
F5A9' 30
55 5A D0
5A D4
51 06 D0
F59E' 30
04 50 E8
5A 04 AB D0
6A 65 OE
5A 55 D0
50 60' D0
56 D5
OF 13
02 E0
0A 0B A6
55 66 OF
00000000'FF 65 OE
05 OA85

```
0A86 2278 .SBTTL NET$SPCINS_NDI - INSERT NDI DATABASE INTO BINARY TREE
0A86 2279 :+
0A86 2280 : NET$SPCINS_NDI - Insert NDI database into binary tree
0A86 2281 :
0A86 2282 : This routine is called to create an entry in the collate and name
0A86 2283 : AVL tree for the new NDI.
0A86 2284 :
0A86 2285 : Inputs:
0A86 2286 :
0A86 2287 :     R11 = Address of CNR
0A86 2288 :     R10 = Address of CNF
0A86 2289 :     R6  = Address of old CNF (or 0 if no old CNF)
0A86 2290 :
0A86 2291 : Outputs:
0A86 2292 :
0A86 2293 :     R0 = Always True.
0A86 2294 :
0A86 2295 :     R5 is destroyed.
0A86 2296 :
0A86 2297 :-
0A86 2298
0A86 2299 NET$SPCINS_NDI::
55 5A D0 0A86 2300      MOVL    R10,R5          ; Insert NDI block
5A 56 D0 0A89 2301      MOVL    R6,R10         ; Save new CNF address
      18 13 0A8C 2302      BEQL    40$         ; Copy old CNF address
00000000'EF 56 D1 0A8E 2303      CMPL    R6,NET$GL_DUM_NDI ; Br if no old CNF
      0F 13 0A95 2304      BEQL    40$         ; Is this the phantom NDI?
      F566' 30 0A97 2305      BSBW    NET$DELETE_BTE ; Br if yes, skip deletion business
55 56 D1 0A9A 2306      CMPL    R6,R5         ; Delete the old BTEs
      07 13 0A9D 2307      BEQL    40$         ; Is new CNF same as old CNF?
00000000'FF 66 0E 0A9F 2308      INSQUE (R6),@NET$GQ_TMP_BUF ; Br if yes, don't deallocate
5A 55 D0 0AA6 2309 40$: MOVL    R5,R10         ; Queue old CNF block for deallocation
      F554' 30 0AA9 2310      BSBW    NET$ADD_NDI   ; Restore new CNF address
50 00' D0 0AAC 2311      MOVL    S^#SS$_NORMAL,R0 ; Insert the new BTEs pointing to CNF
      05 05 0AAF 2312 90$: RSB                ; Indicate success
0AB0 2313                ; Return to caller
```

```
0AB0 2315 .SBTTL NET$DELETE_XXX - PRE-DELETE PROCESSING
0AB0 2316 :+
0AB0 2317 : NET$DELETE_XXX - Special processing before an entry is marked for delete
0AB0 2318 :
0AB0 2319 : This routine is called to perform any special action that may need to be
0AB0 2320 : taken before marking a CNF for delete.
0AB0 2321 :
0AB0 2322 : INPUTS:      R11      CNR pointer
0AB0 2323 :             R10      CNF pointer
0AB0 2324 :
0AB0 2325 : OUTPUTS:     R11,R10 Preserved
0AB0 2326 :             R0       LBS if successful
0AB0 2327 :             LBC if CNF should not be marked for delete
0AB0 2328 :
0AB0 2329 : All registers may be destroyed.
0AB0 2330 :
0AB0 2331 : NET$DELETE_LNI:: : Special processing before marking
0AB0 2332 : NET$DELETE_AJI::
0AB0 2333 : NET$DELETE_SDI::
0AB0 2334 : NET$DELETE_ARI::
50 D4 0AB0 2335 CLRL R0 : ABSOLUTELY NOT
05 0AB2 2336 RSB
0AB3 2337
0AB3 2338 NET$DELETE_NDI:: : CNF for delete
50 D4 0AB3 2339 CLRL R0 : Assume not delete-able
02 E0 0AB5 2340 BBS #CNFSV FLG ACP,- : Not deleteable if the ACP owns
27 0B AA 0AB7 2341 CNFSB FLG(R10),10$ : this block
05 E0 0ABA 2342 BBS #NDI V LOCAL,- : If set then this is the "local" node
1F 0B AA 0ABC 2343 CNFSB FLG(R10),5$ : and cannot be deleted
0306 8F BB 0ABF 2344 PUSHR #M<R1,R2,R8,R9> : Save regs
F52D' 30 0AD0 2345 $GETFLD ndi,s,col : Get the collating value
0306 8F BA 0AD3 2346 BSBW NET$DELETE_BTE : Delete the BTEs for this CNF
00000000'FF 6A 0E 0AD7 2347 POPR #M<R1,R2,R8,R9> : Restore regs
50 01 9A 0ADE 2348 INSQUE (R10),@NET$GQ_TMP_BUF : Insert buffer on TMP_BUF queue.
05 0AE1 2349 5$: MOVZBL #1,R0 : Indicate success
0AE2 2350 10$: RSB
0AE2 2351
0AE2 2352 NET$DELETE_OBI::
50 96 0AEF 2353 $GETFLD obi,l,pid : See if declared name
05 0AF1 2354 INCB R0 : Invert status -- not delete-able if
0AF2 2355 RSB : declared name
0AF2 2356
0AF2 2357 NET$DELETE_ESI::
50 D4 0AFF 2358 $GETFLD esi,v,lck : See if its locked
02 58 E8 0B01 2359 CLRL R0 : Assume not not delete-able
50 96 0B04 2360 BLBS R8,10$ : If locked then not delete-able
05 0B06 2361 INCB R0 : Else its delete-able
0B07 2362 10$: RSB
0B07 2363
0B07 2364 NET$DELETE_EFI::
50 D4 0B14 2365 $GETFLD efi,v,lck : See if its locked
02 58 E8 0B16 2366 CLRL R0 : Assume not not delete-able
50 96 0B19 2367 BLBS R8,10$ : If locked then not delete-able
05 0B1B 2368 INCB R0 : Else its delete-able
0B1C 2369 10$: RSB
0B1C 2370
0B1C 2371 NET$DELETE_LLI:: : Delete Logical-link action routine
```

```
50 24 AA D0 0B1C 2372      MOVL      CNFSC_LENGTH + -      : Get XWB
                                0B20 2373      LLISL_XWB(R10),R0      :
55 00000000'EF 13 0B20 2374      BEQL      10$      : If EQL, then its not there
53 3E A0 3C 0B29 2375      MOVL      NET$GL NET UCB, R5      : Provide a 'NET' UCB address
51 52 08 9A 0B2D 2376      MOVZWL     XWBSW [OCLNK(R0),R3      : Get logical link number
50 3A A0 3C 0B30 2377      MOVZBL     #NET$C DR THIRD, R2      : Disconnect reason
50 09 9A 0B34 2378      MOVZWL     XWBSW REMNOD(R0),R1      : Get partner node address
50 F4C6' 30 0B37 2379      MOVZBL     #NETUPDS DSCLNK, R0      : Setup function code
50 00' D0 0B3A 2380      BSBW      CALL NETDRIVER      : Call the driver
                                05 0B3D 2381 10$: MOVL      S^#SS$ _NORMAL,R0      : Setup status
                                0B3E 2382      RSB      : Done
                                0B3E 2383
                                0B3E 2384 NET$DELETE SPI::      : Delete SPI database entry
50 01 D0 0B3E 2385      MOVL      #1,R0      : No special checking
                                05 0B41 2386      RSB
```

```
OB42 2388 .SBTTL NET$REMOVE_XXX - PROCESS THE REMOVE REQUEST
OB42 2389 .SBTTL NET$REMOVE_DEF - DEFAULT PROCESSING OF THE REMOVE REQUEST
OB42 2390 :+
OB42 2391 : NET$REMOVE_XXX - Processing after a block has been removed.
OB42 2392 : NET$REMOVE_DEF - Default processing of the remove request.
OB42 2393 :
OB42 2394 : This routine is called to perform special processing after a CNF block has
OB42 2395 : been removed from the database. On return, the block is deallocated.
OB42 2396 :
OB42 2397 : INPUTS: R11 CNR pointer
OB42 2398 :
OB42 2399 : OUTPUTS: All registers are preserved.
OB42 2400 :-
OB42 2401 NET$REMOVE_LNI:: ; Remove Local node CNF action routine
OB42 2402 NET$REMOVE_OBI:: ; Remove Object CNF action routine
OB42 2403 NET$REMOVE_SPI::
OB42 2404 NET$REMOVE_AJI::
OB42 2405 NET$REMOVE_SDI::
OB42 2406 NET$REMOVE_ARI::
OB42 2407 NET$REMOVE_DEF:: ; Default CNF removal routine
OB42 2408 PUSHQ R0 ; Save registers
OB45 2409
10 51 5B D0 OB45 2410 MOVL R11,R1 ; Start at head of queue
50 03 11 OB48 2411 BRB 30$ ; Continue
51 61 D0 OB4A 2412 ASSUME CNR$FLINK EQ CNF$FLINK
50 61 D0 OB4A 2413 20$: MOVL CNF$FLINK(R1),R1 ; Advance the pointer
10 50 61 D0 OB4D 2414 30$: MOVL CNF$FLINK(R1),R0 ; Advance to next CNF
OB  A0 00 E0 OB50 2415 BBS #CNF$V_FLG_CNR,CNF$B_FLG(R0),40$ ; If BS then at root -- done
OB 01 E5 OB55 2416 BBCC #CNF$V_FLG_DELETE,- ;
FO OB A0 OB57 2417 CNF$B_FLG(R0),20$ ; If BC not marked for delete
50 60 OF OB5A 2418 ASSUME CNF$FLINK EQ 0
OB5A 2419 REMQUE (R0),R0 ; Remove this entry
OB5D 2420
OB5D 2421 ; Deallocate the CNF block
OB5D 2422
00000000'EF 16 OB5D 2423 JSB NET$DEALLOCATE ; Deallocate the block
E8 11 OB63 2424 BRB 30$
OB65 2425
OB65 2426 40$: POPQ R0 ; Restore registers
05 OB68 2427 RSB
OB69 2428
OB69 2429 NET$REMOVE_LLI:: ; Try to remove LLI entries
OB69 2430 PUSHQ R0 ; Save registers
OB6C 2431
51 5B D0 OB6C 2432 MOVL R11,R1 ; Start at head of queue
50 03 11 OB6F 2433 BRB 30$ ; Continue
51 61 D0 OB71 2434 ASSUME CNR$FLINK EQ CNF$FLINK
50 61 D0 OB71 2435 20$: MOVL CNF$FLINK(R1),R1 ; Advance the pointer
14 50 61 D0 OB74 2436 30$: MOVL CNF$FLINK(R1),R0 ; Advance to next CNF
OB  A0 00 E0 OB77 2437 BBS #CNF$V_FLG_CNR,CNF$B_FLG(R0),40$ ; If BS then at root -- done
OB  A0 02 8A OB7C 2438 BICB #CNF$M_FLG_DELETE,CNF$B_FLG(R0) ; Erase delete marker
24 A0 D5 OB80 2439 TSTL CNF$C_LENGTH+LLI$XWB(R0) ; Still pointing to an XWB ?
OB  EC 12 OB83 2440 BNEQ 20$ ; If NEQ yes, don't remove
50 60 OF OB85 2441 ASSUME CNF$FLINK EQ 0
OB85 2442 REMQUE (R0),R0 ; Remove this entry
OB88 2443
OB88 2444 ; Deallocate the CNF block
```



```
00000000'EF 16 0B88 2445 ;
E4 11 0B88 2446 JSB NET$DEALLOCATE ; Deallocate the block
0B8E 2447 BRB 30$ ;
0B90 2448 ;
0B90 2449 40$: POPQ R0 ; Restore registers
0B93 2450 RSB ;
0B94 2451 ;
0B94 2452 NET$REMOVE_EFI:: ; Remove Event filter CNF action rou
0B94 2453 NET$REMOVE_ESI:: ; Remove Event sink CNF action routi
0B94 2454 PUSHQ R0 ; Save registers
51 5B D0 0B97 2455 MOVL R11,R1 ; Start at head of queue
03 11 0B9A 2456 BRB 30$ ; Continue
51 61 D0 0B9C 2457 ASSUME CNF$FLINK EQ CNF$FLINK ;
50 61 D0 0B9F 2458 20$: MOVL CNF$FLINK(R1),R1 ; Advance the pointer
18 0B A0 00 E0 0BA2 2459 30$: MOVL CNF$FLINK(R1),R0 ; Advance to next CNF
01 E1 0BA7 2460 BBS #CNF$V-FLG-CNR,CNF$B-FLG(R0),40$ ; If BS then at root -- done
F0 0B A0 01 E1 0BA9 2461 BBC #CNF$V-FLG-DELETE,- ;
0BAC 2462 CNF$B-FLG(R0),20$ ; If BC not marked for delete
50 60 0F 0BAC 2463 ASSUME CNF$FLINK EQ 0 ;
0BAF 2464 REMQUE (R0),R0 ; Remove this entry
0BAF 2465 ;
0BAF 2466 ; Deallocate the CNF block
0BAF 2467 ;
50 DD 0BAF 2468 PUSHL R0 ; Save block address
F44C' 30 0BB1 2469 BSBW NET$DBC_EFI ; Inform EVL of database change
50 8ED0 0BB4 2470 POPL R0 ; Restore block address
0BB7 2471 ;
00000000'EF 16 0BB7 2472 JSB NET$DEALLOCATE ; Deallocate the block
E0 11 0BBD 2473 BRB 30$ ;
0BBF 2474 40$: POPQ R0 ; Restore registers
05 0BC2 2475 RSB ;
0BC3 2476 ;
0BC3 2477 NET$REMOVE_NDI:: ; Remove NDI from the list
05 0BC3 2478 RSB ;
```

```
0BC4 2480 .SBTTL SCAN_XWB - SCAN XWB LIST
0BC4 2481 :+
0BC4 2482 : SCAN_XWB - Scan XWB list to total active links and delay
0BC4 2483 :
0BC4 2484 : INPUTS: R11 NDI CNR address
0BC4 2485 : R10 NDI CNF address
0BC4 2486 : R9 scratch
0BC4 2487 : R8 Node address
0BC4 2488 : R7 RCB address
0BC4 2489 : R1 - R0 scratch
0BC4 2490 :
0BC4 2491 : OUTPUTS: R7 Average delay found
0BC4 2492 : R8 Total number of active links
0BC4 2493 :-
0BC4 2494 .SAVE PSECT
00000000 2495 .PSECT NET_LOCK_CODE,NOWRT,GBL
0000 2496
0000 2497 SCAN_XWB:
0000 2498 PUSH R2,R6
51 58 D0 0004 2499 MOV R8,R1
09 12 0007 2500 BNEQ 5$
51 0E A7 3C 0009 2501 MOVZWL RCB$W_ADDR(R7),R1
52 008D C7 3C 000D 2502 MOVZWL RCB$W_ALIAS(R7),R2
0012 2503 5$: DSBINT #NET$IPL
50 24 A7 D0 0018 2504 MOV RCB$L_PTR_LTB(R7),R0
59 04 A0 3C 001C 2505 MOVZWL LTB$W_SLT_TOT(R0),R9
57 0020 2506 CLRQ R7
56 10 A049 D0 0022 2507 10$: MOV LTB$L_SLOTS(R0)[R9],R6
15 56 E8 0027 2508 BLBS R6,20$
51 3A A6 B1 002A 2509 CMPW XWB$W_REMNOD(R6),R1
06 13 002E 2510 BEQL 15$
52 3A A6 B1 0030 2511 CMPW XWB$W_REMNOD(R6),R2
09 12 0034 2512 BNEQ 20$
58 D6 0036 2513 15$: INCL R8
7E 4E A6 3C 0038 2514 MOVZWL XWB$W_DELAY(R6),-(SP)
57 8E C0 003C 2515 ADDL (SP)+,R7
E0 59 F5 003F 2516 20$: SOBGTR R9,10$
0042 2517 ENBINT
50 D4 0045 2518 CLRL R0
58 D5 0047 2519 TSTL R8
06 13 0049 2520 BEQL 30$
57 58 C6 004B 2521 DIVL R8,R7
50 01 D0 004E 2522 MOVL #1,R0
0044 8F BA 0051 2523 30$: POPR #*M<R2,R6>
05 0055 2524 RSB
0056 2525
00000BC4 2526 .RESTORE_PSECT
```

: Scan XWB list
: Save some reg(s)
: Copy the address
: If NEQ then not local
: Copy the local address
: Copy the local alias address
: Synchronize with NETDRIVER
: Get LTB address
: Get no. of XWB's to check
: Init link count and delay
: Get the next XWB
: If LBS slot not in use
: Remote node match?
: Branch if match
: Alias match?
: Branch if not
: Increment total active links
: Get delay
: Add to total
: Increment slot number and loop
: Restore IPL
: Assume node links
: Any links?
: If EQL no
: Compute average delay
: Indicate success
: Restore registers

```
.SBTTL LNI PARAMETER ACTION ROUTINES
OBC4 2528
OBC4 2529
OBC4 2530 :+ NET$LNI_V_LCK - Get status of conditionally writeable fields
OBC4 2531
OBC4 2532 : NET$LNI_L_ADD - Read or write executor address
OBC4 2533 : NET$LNI_L_ACL - Get number of currently active links
OBC4 2534
OBC4 2535 : NET$LNI_S_COL - Get collating value
OBC4 2536 : NET$LNI_S_NAM - Get local node name
OBC4 2537 : NET$LNI_S_CNT - Get (optionally clear) local counters
OBC4 2538 : NET$LNI_S_PHA - Get NI physical address to be used by this node
OBC4 2539
OBC4 2540 : INPUTS: R11 LNI CNR address
OBC4 2541 : R10 LNI CNF address
OBC4 2542 : R9 FLD i.d. of field being read
OBC4 2543 : R1 Scratch
OBC4 2544 : R0 Scratch
OBC4 2545
OBC4 2546 : OUTPUTS: R1 Address of field value or longword string descriptor
OBC4 2547 : R0 Low bit set if R1 is valid
OBC4 2548 : Low bit clear otherwise
OBC4 2549
OBC4 2550 : All other register values are preserved.
OBC4 2551
OBC4 2552 :-
OBC4 2553 NET$LNI_V_LCK::
OBC4 2554 BSBW NET$GET_LOC_STA ; Get status of cond. writeable fields
50 0930 30 OBC7 2555 CMPB #LNISC_STA_OFF,R0 ; Return local state in R0
50 0A 13 OBCA 2556 BEQL 5$ ; Is the ACP off?
50 04 91 OBCC 2557 CMPB #LNISC_STA_INIT,R0 ; If so okay to write fields
50 05 13 OBCF 2558 BEQL 5$ ; Is the ACP init'ing
51 01 90 OBD1 2559 MOVB #1,R1 ; If so okay to write fields
51 02 11 OBD4 2560 BRB ; Indicate fields are locked
51 51 D4 OBD6 2561 5$: CLRL R1 ; Continue
50 00' D0 OBD8 2562 10$: MOVL S^#SS$_NORMAL,R0 ; Fields are not locked
05 OBD8 2563 RSB ; Success
OBC4 2564
OBC4 2565 NET$LNI_L_STA::
26 06 50 E9 OBCD 2566 BCB R0,50$ ; Executor state (read/write parameter)
26 AA 58 90 OBDF 2567 MOVB R8,CNF$C_LENGTH+LNISB_STA(R10) ; Branch if to be read
51 0C 11 OBE3 2568 BRB 80$ ; Store in LNI
51 26 AA 9A OBE5 2569 50$: MOVZBL CNF$C_LENGTH+LNISB_STA(R10),R1 ; Get executor state
51 04 51 91 OBE9 2570 CMPB R1,#LNISC_STA_INIT ; 'initializing' state?
51 03 12 OBEC 2571 BNEQ 80$ ; If not, then ok
51 01 9A OBEE 2572 MOVZBL #NMASC_STATE_OFF,R1 ; Assume OFF
50 00' D0 OBF1 2573 80$: MOVL S^#SS$_NORMAL,R0 ; Return success
05 OBF4 2574 RSB
OBC4 2575
OBC4 2576 NET$LNI_L_ADD::
24 06 50 E9 OBF5 2577 BCB R0,50$ ; Executor address (read/write parameter)
24 AA 58 B0 OBF8 2578 MOVW R8,CNF$C_LENGTH+LNISW_ADD(R10) ; Branch if to be read
51 07 11 OBF0 2579 BRB 80$ ; Store in LNI
51 24 AA 3C OBF6 2580 50$: MOVZWL CNF$C_LENGTH+LNISW_ADD(R10),R1 ; Get node address
51 0333 30 OC02 2581 BSBW SUPPRESS_AREA ; Suppress area, if necessary
50 00' D0 OC05 2582 80$: MOVL S^#SS$_NORMAL,R0 ; Return success
05 OC08 2583 RSB
OC09 2584
```

```
50 00000000'EF D0 0C09 2585 NET$LNI_L_ACL:: ; Get number of currently active links
      09 13 0C09 2586      MOVL NET$GL_PTR_VCB,R0 ; Get RCB pointer
      51 54 A0 3C 0C10 2587      BEQL 10$ ; Br on error
      51 D7 0C12 2588      MOVZWL RCB$W_MCOUNT(R0),R1 ; Get number of links + 1
      50 01 90 0C16 2589      DECL R1 ; Subtract out the ACP reference
      05 0C18 2590      MOVB #1,R0 ; Indicate success
      0C1B 2591 10$: RSB
      0C1C 2592
      83 01 90 0C1C 2593 NET$LNI_S_COL:: ; Get collating value
      50 00'8F 90 0C1C 2594      MOVB #1,(R3)+ ; Enter a static value
      90 0C1F 2595      MOVB #$$$_NORMAL,R0 ; Indicate success
      05 0C23 2596      RSB
      0C24 2597
      5B 00000000'EF D0 0C24 2598 NET$LNI_S_NAM:: ; Get local node name
      5A 00000000'EF D0 0C2B 2599      MOVL NET$GL_CNR_NDI,R11 ; Get the NDI root block
      08 50 E9 0C32 2600      MOVL NET$GL_LOCAL_NDI,R10 ; Get the local NDI CNF
      63 68 57 28 0C3F 2601      $GETFLD ndi,s,na ; Get node name field
      50 00'8F 90 0C42 2602      BLBC R0,10$ ; Branch if not present
      28 0C42 2603      MOVC R7,(R8),(R3) ; Copy into buffer
      90 0C46 2604      MOVB #$$$_NORMAL,R0 ; Indicate success
      05 0C4A 2605 10$: RSB ; Return status in R0
      0C4B 2606
      5B 00000000'EF D0 0C4B 2607 NET$LNI_S_CNT:: ; Get (optionally clear) local counters
      5A 00000000'EF D0 0C52 2608      MOVL NET$GL_CNR_NDI,R11 ; Get the NDI root block
      018B 30 0C59 2609      MOVL NET$GL_LOCAL_NDI,R10 ; Get the local NDI CNF
      05 0C5C 2610      BSBW NET$NDI_S_CNT ; Get the counter block
      0C5D 2611      RSB ; Return status in R0
      0C5D 2612
      0A 50 E9 0C5D 2613 NET$LNI_S_PHA:: ; Get node address
      83 000400AA 8F D0 0C6A 2614      $GETFLD lni,l,add ; Error if not set
      83 58 B0 0C6D 2615      BLBC R0,90$ ; Set NI Phase IV prefix
      05 0C74 2616      MOVL #TR$C_NI_PREFIX,(R3)+ ; Append Phase IV node address
      0C77 2617      MOVW R8,(R3)+ ; Success
      2618 90$: RSB
```

```
OC78 2620 .SBTTL NDI PARAMETER ACTION ROUTINES
OC78 2621 :+
OC78 2622 : NET$NDI_V_REA - Get node reachability status
OC78 2623 : NET$NDI_V_LCK - Get status of conditionally writeable fields
OC78 2624 : NET$NDI_V_LOO - Get bit which is set if the CNF is for a "loopback" nodename
OC78 2625 :
OC78 2626 : NET$NDI_L_ADD - Read or write node address
OC78 2627 : NET$NDI_L_ACL - Get number of active links to the node
OC78 2628 : NET$NDI_L_DEL - Get delay to node
OC78 2629 : NET$NDI_L_DTY - Get node type
OC78 2630 : NET$NDI_L_DCO - Get total cost to node
OC78 2631 : NET$NDI_L_DHO - Get total hops to node
OC78 2632 : NET$NDI_L_TAD - Get transformed node address
OC78 2633 : NET$NDI_L_NND - Get next hop node address on path to remote node
OC78 2634 :
OC78 2635 : NET$NDI_S_HAC - Get merged node address/loopback linename value
OC78 2636 : NET$NDI_S_COL - Get collating sequence value
OC78 2637 : NET$NDI_S_DLI - Get line for normal traffic to node
OC78 2638 : NET$NDI_S_CNT - Get (optionally clear) node counters
OC78 2639 :
OC78 2640 : INPUTS: R11 NDI CNR address
OC78 2641 : R10 NDI CNF address
OC78 2642 : R9 FLD i.d. of field being read
OC78 2643 : R1 Scratch
OC78 2644 : R0 Scratch
OC78 2645 :
OC78 2646 : OUTPUTS: R1 Address of field value or longword string descriptor
OC78 2647 : R0 Low bit set if R1 is valid
OC78 2648 : Low bit clear otherwise
OC78 2649 :
OC78 2650 : All other register values are preserved.
OC78 2651 :
OC78 2652 : -
OC78 2653 : *
OC78 2654 : **
OC78 2655 : *** This set of routines assumes that the node address field ***
OC78 2656 : *** is not an action routine and that it is always set. ***
OC78 2657 : **
OC78 2658 : *
OC78 2659 :
OC78 2660 : NET$NDI_V_LOO:: ; See if CNF is for a loopback node
51 01 D0 OC78 2661 : MOVL #1,R1 ; Assume loop node
02 0B AA E0 OC78 2662 : BBS #NDI_V_LOOP,- ; If BS then loop node
51 51 D4 OC7D 2663 : CNF$B_FLG(R10),10$ ;
50 00' D0 OC80 2664 : CLRL R1 ; Not a loop node
05 05 D0 OC82 2665 : 10$: MOVL S^#SS$ _NORMAL,R0 ; Success
OC85 2666 : RSB ; Done
OC86 2667 :
OC86 2668 : NET$NDI_V_LCK:: ; Get status of cond. writeable fields
51 51 D4 OC86 2669 : CLRL R1 ; Say "not locked"
50 00' D0 OC88 2670 : MOVL S^#SS$ _NORMAL,R0 ; Success
05 05 D0 OC8B 2671 : RSB
OC8C 2672 :
OC8C 2673 : NET$NDI_V_REA:: ; Get node reachability status
0265 30 OC8C 2674 : BSBW NDI_SETUP ; Use common setup
OC8F 2675 :
OC8F 2676 : ; See if we know what the next hop is to the remote node.
```

```

      OC8F 2677      ; If we can't determine the next hop, it is definitely
      OC8F 2678      ; unreachable.
      OC8F 2679      ;
      0106 30 OC8F 2680 BSBW NEXT_HOP_ADJ      ; Locate next hop ADJ
      28 50 E9 OC92 2681 BLBC R0,42$          ; If we don't have a next hop,
      OC95 2682      ; then it is definitely unreachable
      OC95 2683      ;
      OC95 2684      ; Either we know it's reachable, or we are relying on another
      OC95 2685      ; node to determine if it's reachable. If we don't really know,
      OC95 2686      ; then return 'don't know'.
      OC95 2687      ;
57 00000000'EF D0 OC95 2688 MOVL NET$GL_PTR_VCB,R7      ; Get the RCB address
      52 24 AA 3C OC9C 2689 MOVZWL NDI_ADD(R10),R2      ; Get the node address
      0A EF OCA0 2690 EXTZV #TR4$V_ADDR_AREA,-      ; Get the remote area number
      50 52 06 OCA2 2691 #TR4$S_ADDR_AREA,R2,R0
      07 13 OCA5 2692 BEQL 39$      ; If area = 0, then use our area
      008B C7 50 91 OCA7 2693 CMPB R0,RCBSB_HOMEAREA(R7) ; Our area?
      16 12 OCAC 2694 BNEQ 45$      ; If not, we don't know it's status
      07 0B AA 05 E0 OCAE 2695 39$: BBS #NDI_V_LOCAL,CNFSB_FLG(R10),40$ ; Skip check if local NDI
      05 008A C7 91 OCB3 2696 CMPB RCBSB_ETY(R7),#ADJ$C_PTY_PH4N ; Are we an endnode?
      0A 13 OCB8 2697 BEQL 45$      ; If so, skip reachability check
      02B2 30 OCBA 2698 40$: BSBW NET$TEST_REACH      ; Return reachability bit in R0
      51 50 9A OCBD 2699 42$: MOVZBL R0,R1      ; Return as param value
      50 00' D0 OCCU 2700 MOVL S^#SS$_NORMAL,R0      ; Success
      05 05 OCC3 2701 RSB
      50 D4 OCC4 2702      ;
      05 05 OCC6 2703 45$: CLRL R0      ; Failure ('don't know')
      OCC7 2704 RSB
      OCC7 2705      ;
      06 50 E9 OCC7 2706 NET$NDI_L_ADD::      ; Node address (read/write parameter)
      24 AA 58 B0 OCCA 2707 BLBC R0,50$      ; Branch if to be read
      07 11 OCCE 2708 MOVW R8,CNFS$C_LENGTH+NDISW_ADD(R10) ; Store in NDI
      51 24 AA 3C OCD0 2709 BRB 80$
      0261 30 OCD4 2710 50$: MOVZWL CNFS$C_LENGTH+NDISW_ADD(R10),R1 ; Get node address
      50 00' D0 OCD7 2711 BSBW SUPPRESS_AREA      ; Suppress area, if necessary
      05 05 OCDA 2712 80$: MOVL S^#SS$_NORMAL,R0      ; Return success
      2713 RSB
      2714      ;
      2715 NET$NDI_L_DCO::      ; Get total cost to node
      14 10 OCDB 2716 BSBW GET_COST_HOPS      ; Get cost/hops to node
      51 51 05 50 E9 OCDD 2717 BLBC R0,90$      ; Exit if error detected
      0A 00 EF OCE0 2718 EXTZV #0,#10,R1,R1      ; Get cost
      05 05 OCE5 2719 90$: RSB
      2720      ;
      2721 NET$NDI_L_DHO::      ; Get total hops to node
      09 10 OCE6 2722 BSBW GET_COST_HOPS      ; Get cost/hops to node
      51 51 05 50 E9 OCE8 2723 BLBC R0,90$      ; Exit if error detected
      0A 0A EF OCEB 2724 EXTZV #10,#5,R1,R1      ; Get hops
      05 05 OCF0 2725 90$: RSB
      2726      ;
      2727 GET_COST_HOPS:      ; Get cost/hops to node
      0200 30 OCF1 2728 BSBW NDI_SETUP      ; Call common setup
      58 D5 OCF4 2729 TSTL R8      ; Address 0?
      06 12 OCF6 2730 BNEQ 5$      ; If so,
      58 0E A7 3C OCF8 2731 MOVZWL RCBSW_ADDR(R7),R8      ; then use our local address
      07 11 OCFC 2732 BRB 8$      ; and skip following endnode check
      05 008A C7 91 OCFE 2733 5$: CMPB RCBSB_ETY(R7),#ADJ$C_PTY_PH4N ; Are we an endnode?
```

```

      3B 13 OD03 2734 BEQL 90$ : If so, then cost/hops 'not known'
      0A EF OD05 2735 8$: EXTZV #TR4$V_ADDR_AREA,- : Get the area number
54 58 06 OD07 2736 : #TR4$$_ADDR_AREA,R8,R4
      07 13 OD0A 2737 BEQL 10$ : If area = 0, assume our area
008B C7 54 91 OD0C 2738 CMPB R4,RCB$B_HOMEAREA(R7) : Our area?
      2D 12 OD11 2739 BNEQ 90$ : If not, then 'not known'
      00 EF OD13 2740 10$: EXTZV #TR4$V_ADDR_DEST,- : Get the node number within area
55 58 0A OD15 2741 : #TR4$$_ADDR_DEST,R8,R5
      5A A7 55 B1 OD18 2742 CMPW R5,RCB$W_MAX_ADDR(R7) : Is node within range?
      22 1A OD1C 2743 BGTRU 90$ : If GTRU then no
      01E4 30 OD1E 2744 BSBW TEST_REACH : Get the node's type
      11 50 E9 OD21 2745 BLBC R0,50$ : If we don't have it, not Phase II
50 51 FO 8F 78 OD24 2746 ASHL #16,R1,R0 : Isolate node type code
      02 50 B1 OD29 2747 CMPW R0,#ADJ$C_PTY_PH2 : Phase II direct adjacency?
      07 12 OD2C 2748 BNEQ 50$ : Branch if not
51 0400 8F 3C OD2E 2749 MOVZWL #1010+0,R1 : Return cost=0, hops=1
      08 11 OD33 2750 BRB 80$ : Exit with success
      OD35 2751
51 00000000'GF45 3C OD35 2752 50$: MOVZWL G^NET$AW_MIN_C_H[R5],R1 : Get cost/hops to node
      50 00' D0 OD3D 2753 80$: MOVL S^N$S$_NORMAL,R0 : Success
      05 OD40 2754 90$: RSB
      OD41 2755
      OD41 2756 NET$NDI_L_ACL:: : Get number of active links
      01B0 30 OD41 2757 BSBW NDI_SETUP : Call common setup
00000000'EF 16 OD44 2758 JSB SCAN_XWB : Scan XWB list and total up links
51 58 D0 OD4A 2759 MOVL R8,RT : Store number of links
      05 OD4D 2760 RSB
      OD4E 2761
      OD4E 2762 NET$NDI_L_DEL:: : Get delay to node
      01A3 30 OD4E 2763 BSBW NDI_SETUP : Call common setup
00000000'EF 16 OD51 2764 JSB SCAN_XWB : Scan XWB list and get average delay
51 57 D0 OD57 2765 MOVL R7,RT : Store average delay
      05 OD5A 2766 RSB
      OD5B 2767
      OD5B 2768 NET$NDI_L_DTY:: : Get node type
      0196 30 OD5B 2769 BSBW NDI_SETUP : Call common setup
      01A4 30 OD5E 2770 BSBW TEST_REACH : See if node is reachable
51 51 10 50 E9 OD61 2771 BLBC R0,30$ : If LBC then no
51 51 51 9C OD64 2772 ROTL #16,R1,R1 : We only want the type
51 FFFF 8F 3C OD68 2773 MOVZWL R1,R1 : Extract off the type
      02 12 OD70 2775 BNEQ 30$ : Is the type 'unknown'?
      50 D4 OD72 2776 CLRL R0 : If not return LBS in R0
      05 OD74 2777 30$: RSB : Else indicate failure
      OD75 2778
      OD75 2779 NET$NDI_L_TAD:: : Get node transformed address
      017C 30 OD75 2780 BSBW NDI_SETUP : Call common setup
      51 58 D0 OD78 2781 NDI_L_TAD: : Get node transformed address
      04 12 OD78 2782 MOVL R8,R1 : Copy the address for returned value
51 0E A7 3C OD7D 2783 BNEQ 10$ : If NEQ then not local
      01B4 30 OD81 2784 MOVZWL RCB$W_ADDR(R7),R1 : Copy the local address
50 01 90 OD84 2785 10$: BSBW SUPPRESS_AREA : Suppress area if necessary
      05 OD87 2786 MOVB #1,R0 : Indicate success
      OD88 2787 RSB
      OD88 2788
      OD88 2789 NET$NDI_L_NND:: : Call common setup
      0169 30 OD88 2790 BSBW NDI_SETUP
```

```
51 07 08 10 0D8B 2791 BSBB NEXT_HOP_ADJ ; Locate next hop ADJ
    04 A7 09 0D8D 2792 BLBC R0,30$ ; If LBC then no
    01A1 3C 0D90 2793 MOVZWL ADJ$W_PNA(R7),R1 ; Get partner's node address
    30 0D94 2794 BSBB SUPPRESS_AREA ; Suppress area if necessary
    05 0D97 2795 30$: RSB
      0D98 2796
      0D98 2797
      0D98 2798 ; Locate next hop ADJ
      0D98 2799
      0D98 2800 ; Inputs:
      0D98 2801
      0D98 2802 R10 = CNF address
      0D98 2803 R8 = Node address
      0D98 2804 R7 = RCB address
      0D98 2805
      0D98 2806 ; Outputs:
      0D98 2807
      0D98 2808 R7 = ADJ address
      0D98 2809 R1 = ADJ index
      0D98 2810 R0 = status
      0D98 2811
      0D98 2812 ; R8 is destroyed.
      0D98 2813
      0D98 2814 NEXT_HOP_ADJ:
      0D98 2815 BBS #NDI_V_LOCAL,- ; If not local node,
      0D9A 2816 CNF$B_FLG(R10),1$
      0D9D 2817 CMPB RCB$B_ETY(R7),#ADJ$C_PTY_PH4N ; Are we an endnode?
      0DA2 2818 BNEQ 1$ ; Skip if not
      0DA4 2819 MOVZWL RCB$W_DRT(R7),R1 ; Setup ADJ to designated router
      0DA9 2820 BRB 8$ ; Get next hop ADJ
      0DAB 2821 1$: EXTZV #TR4$V_ADDR_AREA,- ; Get the area number
      0DAD 2822 #TR4$S_ADDR_AREA,R8,R0
      0DB0 2823 BEQL 10$ ; If area = 0, then use our area
      0DB2 2824 CMPB R0,RCB$B_HOMEAREA(R7) ; Is this in our area?
      0DB7 2825 BEQL 10$ ; If so, then it's ok
      0DB9 2826 CMPB RCB$B_ETY(R7),#ADJ$C_PTY_AREA ; Do we know about outside areas?
      0DBE 2827 BEQL 5$ ; If so, lookup next hop for area
      0DC0 2828 4$: MOVZWL RCB$W_LVL2(R7),R1 ; Get ADJ to nearest level 2 router
      0DC5 2829 BRB 8$ ; Get next hop ADJ
      0DC7 2830 5$: BBC #RCB$V_LVL2,- ; If we are not enabled for Level 2
      0DC9 2831 RCB$B_STATUS(R7),4$ ; routing, then act like a Level 1 router
      0DCC 2832 R0,RCB$B_MAX_AREA(R7) ; Within range?
      0DD1 2833 BGTRU 70$ ; If out of range, unreachable
      0DD3 2834 MOVZWL @RCB$L_PTR_AOA(R7)[R0],R1 ; Get ADJ to area router
      0DD8 2835 8$: MOVL R1,R8 ; Pass ADJ index argument
      0DDB 2836 BSBB NET$FIND_ADJ ; Get next hop ADJ address
      0DDE 2837 BRB 20$ ; Return with R0/R1/R7 set
      0DE0 2838
      0DE0 2839 10$: BSBB TEST_REACH ; See if node is reachable & get ADJ
      0DE3 2840 20$: RSB
      0DE4 2841
      0DE4 2842 70$: CLRL R0 ; Unreachable
      0DE6 2843 RSB
      0DE7 2844
      0DE7 2845 NET$NDI_S_CNT:: ; Get node counters
      0DE7 2846 MOVZWL #SS$_BADPARAM,R0 ; Assume loop-node
      0DEC 2847 BBS #NDI_V_LOOP,- ; If BS then loop node
```



```
54      24 0B AA      ODEE 2848      CNFSB_FLG(R10),30$      : ...no counters for loop nodes
      00000000'EF      DO      ODF1 2849      MOVL      NET$GL_PTR_VCB,R4      : Get RCB address
      53      DD      ODF8 2850
      06'AF      9F      ODF8 2851      PUSH      R3      : Save original output pointer
      12 AA      3C      ODF8 2852      PUSHAB   B^20$      : Setup return address
      13      13      ODFD 2853      MOVZWL   CNFSW_ID(R10),R8      : Get node address
      0040      31      OE01 2854      BEQL      LOCAL_NODE_CNT      : If EQL then local node
      52 8ED0      0E03 2855      BRW       NODE_CNT      : Else remote node
      09 50      E9      OE06 2856      POPL      R2      : Recover original counter block ptr
      50 00      DO      OE09 2857      :
      07CB      30      OE09 2858      BLBC      R0,30$      : Br on error
      50 01      DO      OE0C 2859      MOVL      S^#EVC$C_SRC_NOD,R0      : Setup event database i.d.
      05      05      OE0F 2860      BSBW      LOG_COUNTERS      : Log the counter block if needed
      30$      05      OE12 2861      MOVL      #1,R0      : Success
      30$      05      OE15 2862      RSB       : Done
      05      05      OE16 2863
      05      05      OE16 2864      LOCAL_NODE_CNT:
      05      05      OE16 2865      :
      05      05      OE16 2866      : Get local node counters. First get the common node counters, the
      05      05      OE16 2867      : first block of which is the 'seconds since last zeroed' counter.
      05      05      OE16 2868      : Append the local-node-only counters, the first block of which is
      05      05      OE16 2869      : also the 'seconds since last zeroed' counter. Shift the counters
      05      05      OE16 2870      : to squeeze out this redundant counter.
      05      05      OE16 2871
      58 0E A4      3C      OE16 2872      MOVZWL   RCB$W_ADDR(R4),R8      : Get local node address
      2A      10      OE1A 2873      BSBW      NODE_CNT      : Get common node counters
      05      05      OE1C 2874
      05      05      OE1C 2875      : Now get snap-shot of local node counters from RCB
      05      05      OE1C 2876
      51 0090 C4      9E      OE1C 2877      MOVAB     RCB$L_ABS_TIM(R4),R1      : Point to start of local node
      52 10      3C      OE21 2878      : counters
      05      05      OE21 2879      MOVZWL   #RCB$C_CNT_SIZE+4,R2      : Total size of block
      05      05      OE24 2880
      05      05      OE24 2881      : Now format the counters
      05      05      OE24 2882
      55 0000007C'EF      9E      OE24 2883      MOVAB     RCB_CNT_TAB,R5      : Point to counter formatting table
      57 53      DO      OE2B 2884      MOVL      R3,R7      : Save output buffer pointer
      00000056'EF      16      OE2E 2885      JSB       MOVE_FMT_CNT      : Move and format the counters
      52 53 57      C3      OE34 2886      SUBL      R7,R3,R2      : Get number of bytes just moved
      52 04      C2      OE38 2887      SUBL      #4,R2      : Account for superfluous 'seconds
      05      05      OE3B 2888      : since last zeroed'
      67 04 A7      05      19      OE3B 2889      BLSS      60$      : If LSS then no NDI counts were moved
      52 50      28      OE3D 2890      MOVCL    R2,4(R7),(R7)      : Shift the counters, update R3
      01      90      OE42 2891      MOV      #1,R0      : Indicate success
      05      05      OE45 2892      RSB       : Return status to co-routine
      05      05      OE46 2893
      05      05      OE46 2894      NODE_CNT:
      5E 0050 8F      BB      OE46 2895      : Move common node counters
      00000064 8F      C2      OE4A 2896      PUSHR     #^M<R4,R6>      : Save regs
      56 5E      DO      OE4A 2897      SUBL      #CNT_FMT_BUFSIZ,SP      : Create work area on stack
      54      D4      OE51 2898      MOVL      SP,R6      : Point to it with R6
      02      E1      OE54 2899      CLRL      R4      : Assume we don't clear counters
      02 00000000'EF      0E56 2900      BBC       #NET$V_CLR_CNT,-      : If BC, don't clear the counters
      54      D6      OE58 2901      :
      F19D'      30      OE5E 2902      INCL      R4      : Else, zero the counters
      1C 50      E9      OE60 2903      BSBW      NET$READ_NDI_CNT      : Get the node counters
      52 1C      3C      OE63 2904      BLBC      R0,50$      : Br if no node counters
      52 1C      3C      OE66 2904      MOVZWL   #NDC$C_LENGTH,R2      : Get size of node counter area
```

```
55 00000058'EF 9E 0E69 2905 MOVAB NDC_CNT_TAB,R5 ; Set counter for routing table
    0000009B'EF 16 0E70 2906 JSB FMT_CNT ; Format the counters
    0E76 2907 30$: ;
    0E76 2908 ; Done, restore the stack and return
    0E76 2909 ;
5E 00000064 8F C0 0E76 2910 ADDL #CNT_FMT_BUFSIZ,SP ; Create work area on stack
    0050 8F BA 0E7D 2911 POPR #^M<R4,R8> ; Restore regs
    05 0E81 2912 RSB ; Done, return status in R0
    0E82 2913
50 0000'8F 3C 0E82 2914 50$: MOVZWL #SS$_NOSUCHNODE,R0 ; Node counters unavailable
    ED 11 0E87 2915 BRB 30$ ; And leave
    0E89 2916
    0E89 2917
    0E89 2918 NET$NDI_S_DLI:: ; Get line for normal traffic to node
    0E89 2919 $CNFFLD ndi,s,nli,R9 ; Identify loopback linename
    1F 0B AA E0 0E90 2920 BBS #NDI_V_LOOP,- ; If BS then loopback node
    005C 30 0E92 2921 CNFSB_FLG(R10),5$ ;
    FEFD 30 0E95 2922 BSBW NDI_SETUP ; Call standard setup
    25 50 E9 0E98 2923 BSBW NEXT_HOP_ADJ ; Get next hop ADJ address
    58 51 3C 0E9B 2924 BLBC R0,10$ ; Skip if not found
    F15C' 30 0E9E 2925 MOVZWL R1,R8 ; Setup ADJ index for subroutine call
    1C 50 E9 0EA1 2926 BSBW NET$ADJ_LPD_CRI ; Get the CRI for this adjacency
    50 D4 0EA4 2927 BLBC R0,10$ ; If LBC then index is invalid
    5A D5 0EA7 2928 CLRL R0 ; Assume no CRI
    16 13 0EA9 2929 TSTL R10 ; Is there an associate CRI?
    0EAB 2930 BEQL 10$ ; If EQL no (could be 'local' LPD)
    0EAD 2931 $CNFFLD cri,s,nam,R9 ; Identify the line name field
    F149' 30 0EB4 2932 5$: BSBW CNF$GET_FIELD ; Get it
    09 50 E9 0EB7 2933 BLBC R0,10$ ; Branch if not present
    63 68 57 28 0EBA 2934 MOVC R7,(R8),(R3) ; Copy into buffer
    50 0000'8F 3C 0EBE 2935 MOVZWL #SS$_NORMAL,R0 ; Indicate successful
    05 0EC3 2936 10$: RSB ; Return status in R0
    0EC4 2937
    0EC4 2938
    0EC4 2939
    0EC4 2940
    0EC4 2941
    0EC4 2942
    0EC4 2943
    0EC4 2944
    0EC4 2945
    0EC4 2946
    0EC4 2947
    0EC4 2948
    0EC4 2949
    0EC4 2950
    0EC4 2951
    0EC4 2952
    0EC4 2953 NET$NDI_S_COL:: ; Get collating sequence string
    83 94 0EC4 2954 CCRB (R3)+ ; Begin with lowest possible value
    04 04 E1 0EC6 2955 BBC #NDI_V_LOOP,- ; If BC then not loop node
    FF A3 01 8E 0EC8 2956 CNFSB_FLG(R10),10$ ;
    83 25 AA 90 0ECB 2957 MNEGB #1,-1(R3) ; Start string with huge value
    83 24 AA 90 0ECF 2958 10$: MOVB NDI_ADD+1(R10),(R3)+ ; Enter high order node address byte
    0D 10 0ED7 2959 MOVB NDI_ADD+0(R10),(R3)+ ; Enter low order node address byte
    06 E1 0ED9 2960 BSBB HACT ; Append loopback linename, if any
    0ED9 2961 BBC #NDI_V_MARKER,- ; If marker CNF,
```

```
03 08 AA      0EDB 2962      CNFSB FLG(R10),20$
83 01 90 OEDE 2963      MOVB #1,(R3)+      ; Then add a byte to make the node
      OEE1 2964      ; number unique from normal CNFs
      05 OEE1 2965 20$: RSB
      OEE2 2966      NET$NDI_S_HAC::      ; Get hashed node addr/loopback
      OEE2 2967      ; Linename value
      OEE2 2968      ;
      OEE2 2969      ; This value is used in a uniqueness check to enforce the rule
      OEE2 2970      ; that 'no two NDI entries that have the same node address
      OEE2 2971      ; will be associated with the same loopback circuit name'.
      OEE2 2972      ;
      OEE2 2973      ;
      83 24 AA 80 OEE2 2974      MOVW NDI_ADD(R10),(R3)+      ; Store the node address
      OEE6 2975 HAC1: $CNFFLD ndi,s,nli,R9      ; Identify loopback linename field
      50 06E0 30 OEE6 2976      BSBW MOVSTR      ; Append it to address
      90 01 90 OEE0 2977      MOVW #1,R0      ; Indicate success if we got this far
      05 OEF3 2978      RSB      ; Retrun to co-routine
      OEF4 2979      ;
      OEF4 2980 NDI_SETUP:
      57 58 24 AA 3C OEF4 2981      MOVZWL NDI_ADD(R10),R8      ; Get the address
      50 00000000'EF D0 OEF8 2982      MOVL NET$GL_PTR_VCB,R7      ; Get the RCB address
      50 0000'8F 3C OEFF 2983      MOVZWL #SS$_NDSUCHNODE,R0      ; Assume failure
      05 OF04 2984      RSB
      OF05 2985      ;
      OF05 2986 TEST_REACH:
      52 52 DD OF05 2987      PUSHL R2      ; Save reg
      52 58 DD OF07 2988      MOVL R8,R2      ; Copy address
      0062 30 OF0A 2989      BSBW NET$TEST_REACH      ; Make test
      52 8ED0 OF0D 2990      POPL R2
      05 OF10 2991      RSB
      OF11 2992      ;
      OF11 2993 NET$NDI_S_NNN::      ; Name of next node to destination
      FE74 30 OF11 2994      BSBW NET$NDI_L_NND      ; Get node number of 'next node'
      20 50 E9 OF14 2995      BLBC R0,90$      ; Skip if failure
      58 51 D0 OF17 2996      MOVL R1,R8      ; Set node number to lookup
      0608 30 OF1A 2997      BSBW NET$NDI_BY_ADD      ; Lookup NDI entry
      17 50 E9 OF1D 2998      BLBC R0,90$      ; If not found, then no name
      OF20 2999      $GETFLD ndi,s,nna      ; Get node name field
      07 50 E9 OF2D 3000      BLBC R0,90$      ; Branch if not present
      63 68 57 28 OF30 3001      MOVC R7,(R8),(R3)      ; Copy into buffer
      50 00' 3C OF34 3002      MOVZWL S^#SS$_NORMAL,R0      ; Success
      05 OF37 3003 90$: RSB
```

```
OF38 3005 .SBTTL SUPPRESS_AREA - Suppress area from node address
OF38 3006 :+
OF38 3007 : SUPPRESS_AREA - Suppress area from node address, if necessary
OF38 3008 :
OF38 3009 : For compatibility with older versions of DECnet at the network
OF38 3010 : management layer, we must provide a mechanism so that area numbers
OF38 3011 : are not returned to network management (NML or EVL) if the user
OF38 3012 : doesn't know about areas. This is done by setting a flag when
OF38 3013 : the executor address is set, based on whether the user explicitly
OF38 3014 : set the executor to a specific area or not.
OF38 3015 :
OF38 3016 : This routine expects to be called from a parameter action routine,
OF38 3017 : where NET$V_INTRNL has been set up properly.
OF38 3018 :
OF38 3019 : Inputs:
OF38 3020 :
OF38 3021 : R1 = Node address
OF38 3022 :
OF38 3023 : Outputs:
OF38 3024 :
OF38 3025 : R1 = New node address, with area possibly suppressed
OF38 3026 :
OF38 3027 : All other registers are preserved.
OF38 3028 :-
OF38 3029 :
OF38 3030 SUPPRESS AREA::
05 OF38 3031 RSB ; Just return for now
BB OF39 3032 PUSHF ; Save registers
D0 OF3D 3033 MOVL NET$GL_CNR_LNI,R11 ; Set CNR address
D0 OF44 3034 MOVL NET$GL_PTR_LNI,R10 ; Set CNF address
13 OF4B 3035 BEQL 90$ ; If none, then address must be 0
OF4D 3036 $GETFLD lni,v,sup ; If areas not suppressed,
E9 OF5A 3037 BLBC R8,90$ ; then skip it
E0 OF5D 3038 BBS #NET$V_INTRNL,NET$GL_FLAGS,90$ ; If internal, do not suppress
F0 OF65 3039 INSV #0,#TR4$V_ADDR_AREA,- ; Suppress the area number
OF68 3040 #TR4$S_ADDR_AREA,R1
BA OF6A 3041 90$: POPR #*M<R0,R7,R8,R9,R10,R11> ; Restore registers
05 OF6E 3042 RSB
```

OF81 8F
5B 00000000'EF
5A 00000000'EF
1D
OD 58
05 00000000'EF 09
0A 00
51 06
OF81 8F

```
OF6F 3044 .SBTTL NET$TEST_REACH - Test node reachability
OF6F 3045 :+
OF6F 3046 : NET$TEST_REACH - Test node reachability
OF6F 3047 :
OF6F 3048 : This routine tests the reachability of a node independent of the
OF6F 3049 : presence of a NDI data block. Note that a Phase III non-adjacent
OF6F 3050 : node can be reached even though there is no NDI block for that node
OF6F 3051 : since the output line is mapped by address -- no other information
OF6F 3052 : is required.
OF6F 3053 :
OF6F 3054 : This routine returns:
OF6F 3055 :
OF6F 3056 : 1) whether it is reachable or not
OF6F 3057 : 2) the ADJ to get to the node
OF6F 3058 : 3) the 'node type' (only if node is a direct adjacency)
OF6F 3059 :
OF6F 3060 : Inputs: R2 Node address
OF6F 3061 : R1 Scratch
OF6F 3062 : R0 Scratch
OF6F 3063 :
OF6F 3064 : Outputs: R7 ADJ address
OF6F 3065 : R1 Adjacency index of path used to reach the node
OF6F 3066 : High word has node type
OF6F 3067 : R0 Status
OF6F 3068 :
OF6F 3069 : All other registers are preserved.
OF6F 3070 :-
OF6F 3071 :
OF6F 3072 NET$TEST_REACH::
OF6F 3073 : PUSHB #*M<R3,R4,R6,R8> ; Test for node reachability
OF6F 3074 : PUSHB #0 ; Save registers
OF6F 3075 : MOVZWL #SS$ NOSUCHNODE,R0 ; Init storage on stack
OF6F 3076 : MOVL NET$GL_PTR_VCB,R1 ; Assume node out of range
OF6F 3077 : EXTZV #TR4$V_ADDR_AREA,- ; Get the RCB
OF6F 3078 : #TR4$S_ADDR_AREA,R2,R3 ; Get the area number
OF6F 3079 : BNEQ 5$ ; If area = 0,
OF6F 3080 : MOVZBL RCB$B_HOMEAREA(R1),R3 ; then use our area
OF6F 3081 : EXTZV #TR4$V_ADDR_DEST,- ; Get the node number within area
OF6F 3082 : #TR4$S_ADDR_DEST,R2,R4
OF6F 3083 : CMPB R3,RCB$B_HOMEAREA(R1) ; Is this in our area?
OF6F 3084 : BNEQ 100$ ; If not, then return unreachable
OF6F 3085 : CMPW R4,RCB$W_MAX_ADDR(R1) ; Within range ?
OF6F 3086 : BGTRU 100$ ; If GTRU then out of range
OF6F 3087 : MOVZWL @RCB$B_PTR_OA(R1)[R4],(SP) ; Get ADJ index to the node
OF6F 3088 : MOVL (SP),R8 ; Get index
OF6F 3089 : BSBW NET$FIND_ADJ ; Get ADJ & LPD address
OF6F 3090 : BLBC R0,80$ ; If error, report node unreachable
OF6F 3091 : MOVW #ADJ$C_PTY_UNK,2(SP) ; Assume type is unknown
OF6F 3092 : EXTZV #TR4$V_ADDR_AREA,- ; Get the area of the adjacency
OF6F 3093 : #TR4$S_ADDR_AREA,ADJ$W_PNA(R7),R0
OF6F 3094 : BEQL 10$ ; If area = 0, skip area match
OF6F 3095 : CMPL R0,R3 ; Does area match?
OF6F 3096 : BNEQ 50$ ; If not, then not adjacent
OF6F 3097 : CMPZV #TR4$V_ADDR_DEST,- ; Does node number match?
OF6F 3098 : #TR4$S_ADDR_DEST,ADJ$W_PNA(R7),R4
OF6F 3099 : BNEQ 50$ ; If not, we don't know the type
OF6F 3100 : MOVZBW ADJ$B_PTYPE(R7),2(SP) ; Store type of partner node
```

51 0158 8F BB OF6F 3073
50 0000 8F DD OF73 3074
00000000 EF 3C OF75 3075
53 52 06 DO OF7A 3076
53 008B C1 EF OF81 3077
54 52 0A OF83 3078
008B C1 53 9A OF86 3079
5A A1 54 12 OF88 3080
6E 1C B144 9A OF8D 3081
58 6E DO OF8F 3082
F056 30 91 OF92 3083
25 50 E9 OF97 3084
02 AE FFFF 8F B0 OF99 3085
50 04 A7 06 EF OF9D 3086
53 50 D1 OF9F 3087
0D 12 OFA4 3088
00 ED OFA7 3089
54 04 A7 0A OFAA 3090
02 AE 01 A7 9B OFAD 3091
05 12 OFB3 3092
0A 13 OFB5 3093
05 D1 OFB9 3094
50 0D OFBB 3095
00 12 OFBE 3096
0A OFC0 3097
05 12 OFC2 3098
0A 9B OFC6 3099
01 A7 9B OFC8 3100

NETCNFACT
V04-000

I 12
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00
NET\$TEST_REACH - Test node reachability 5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 Page 67
(33)

50	00'	D0	OFCD	3101	50\$:	MOVL	S^#SS\$ _NORMAL,R0	:	Indicate reachable	
	05	11	OFD0	3102		BRB	100\$	:	And exit with success	
50	0000'	8F	3C	OFD2	3103	80\$:	MOVZWL	#SS\$ _UNREACHABLE,R0	:	Node is known, but unreachable
	51	8ED0	OFD7	3104	100\$:	POPL	R1	:	Recover path and type	
0158	8F	BA	OFDA	3105		POPR	#^M<R3,R4,R6,R8>	:	Restore registers	
		05	OFDE	3106		RSB		:		

NE
VO

```
.SBTTL OBI PARAMETER ACTION ROUTINES

OFDF 3108 :+
OFDF 3109 :+
OFDF 3110 : NET$OBI_V_LCK - Get status of conditionally writeable fields
OFDF 3111 :
OFDF 3112 : NET$OBI_S_COL - Get collating value
OFDF 3113 : NET$OBI_S_ZNA - Get combined number,name
OFDF 3114 : NET$OBI_S_IAC - Get default inbound access
OFDF 3115 : NET$OBI_S_SFI - Startup file id string
OFDF 3116 :
OFDF 3117 : INPUTS:      R11      NDI CNR address
OFDF 3118 :              R10      NDI CNF address
OFDF 3119 :              R9       FLD i.d. of field being read
OFDF 3120 :              R1       Scratch
OFDF 3121 :              R0       Scratch
OFDF 3122 :
OFDF 3123 : OUTPUTS:    R1       Address of field value or longword string descriptor
OFDF 3124 :              R0       Low bit set if R1 is valid
OFDF 3125 :              Low bit clear otherwise
OFDF 3126 :
OFDF 3127 : All other register values are preserved.
OFDF 3128 :
OFDF 3129 : -
OFDF 3130 :
OFDF 3131 : NET$OBI_V_LCK::                                ; Get status of cond. writeable fields
OFDF 3132 :
OFDF 3133 :      If the UCB field is active then the object is a 'declared' object
OFDF 3134 :      or name and the conditionally writeable fields are locked.
OFDF 3135 :
OFDF 3136 : $GETFLD obi,l,ucb                                ; Fetch UCB field
51 50 D0 OFEC 3137 MOVL R0,R1                                ; If UCB field is active then CNF is
OFDF 3138 : locked
50 00' D0 OFEF 3139 MOVL S^#SS$_NORMAL,R0                    ; Success
OFDF 3140 : RSB
OFDF 3141 : Return to co-routine
OFDF 3142 : NET$OBI_S_COL::                                ; Get collating value
OFDF 3143 : NET$OBI_S_ZNA::                                ; Get combined number,name
OFDF 3144 :
OFDF 3145 : This string is used for collating and uniqueness checking. The
OFDF 3146 : blocks are to be collated by number and all non-zero numbers must
OFDF 3147 : be unique (not two CNFs may share a object number) unless that
OFDF 3148 : number is zero -- since declared names will all have the object
OFDF 3149 : type zero. Thus the value of this field is the object number
OFDF 3150 : alone if the number is non-zero, or the object number followed
OFDF 3151 : by the name if the number is zero.
OFDF 3152 :
OFDF 3153 : Note: If the name itself is required to be unique then that
OFDF 3154 : check must be made elsewhere.
OFDF 3155 :
OFDF 3156 : $GETFLD obi,l,num                                ; Get the object number
18 50 E9 1000 3157 BLBC R0,20$                                ; Br on error
83 58 90 1003 3158 MOVB R8,(R3)+                            ; Move the number
13 12 1006 3159 BNEQ 20$                                    ; If NEQ then we're done
OFDF 3160 : $CNFFLD obi,s,nam,R9                            ; Identify the object name
OFDF 3161 : BSBW MOVSTR                                          ; Append it to buffer
05BE 30 100F 3161
OFDF 3162 : BLBC R0,20$                                ; Br on error
06 50 E9 1012 3162
OFDF 3163 : TSTW R7                                          ; Is the length zero ?
OFDF 3164 : BNEQ 20$                                    ; If NEQ then okay
```

```

      50      D4      1019      3165      CLRL      R0      ; Else illegal ZNA value
      05      101B      3166      20$:      RSB      ; Return status to co-routine
      101C      3167
      101C      3168
      03 A3      9F      101C      3169      NET$OBI_S_IAC::      ; Get default inbound access
      101F      3170      -POSHAR      3(R3)      ; Null access strings are 3 null bytes
      059D      30      1026      3171      $CNFFLD      obi,s,usr,R9      ; Setup field id
      1029      3172      BSBW      MOVCSF      ; Move the username
      0593      30      1030      3173      $CNFFLD      obi,s,psw,R9      ; Setup password field id
      1033      3174      BSBW      MOVCSF      ; Move it
      0589      30      103A      3175      $CNFFLD      obi,s,acc,R9      ; Setup account id
      53      8E      D1      103D      3176      BSBW      MOVCSF      ; Move it
      35      12      1040      3177      CMPL      (SP)+,R3      ; Is the access control null?
      7E      5A      7D      1042      3178      BNEQ      10$      ; If NEQ no - proceed
      00000000'EF      D0      1045      3179      MOVQ      R10,-(SP)      ; Save OBI CNF,CNR
      00000000'EF      D0      104C      3180      MOVL      NET$GL_CNR_NDI,R11      ; Get the NDI root block
      53      03      C2      104C      3181      MOVL      NET$GL_LOCAL_NDI,R10      ; Get the local NDI CNF
      1053      3182      SUBL      #3,R3      ; Reset R3
      1056      3183      $CNFFLD      ndi,s,nus,R9      ; Setup field id
      0566      30      105D      3184      BSBW      MOVCSF      ; Move the username
      1060      3185      $CNFFLD      ndi,s,npw,R9      ; Setup password field id
      055C      30      1067      3186      BSBW      MOVCSF      ; Move it
      106A      3187      $CNFFLD      ndi,s,nac,R9      ; Setup account id
      0552      30      1071      3188      BSBW      MOVCSF      ; Move it
      5A      8E      7D      1074      3189      5$:      MOVQ      (SP)+,R10      ; Restore OBI CNF,CNR
      50      01      D0      1077      3190      10$:      MOVL      #1,R0      ; Always successful
      05      107A      3191      RSB      ; Return to co-routine to return desc.
      107B      3192
      107B      3193      NET$OBI_S_SFI::      ; Startup file id string
      107B      3194
      107B      3195      ; Build .COM filename spec for image activation.
      107B      3196
      107B      3197      filename = SYS$SYSTEM:file.COM if the object number NEQ 0
      107B      3198      ; or if object name starts with '$'
      107B      3199      file ; if the object number EQL 0
      107B      3200
      107B      3201      where "file" comes from OBI,S,FID if its defined
      107B      3202      or OBI,S,NAM otherwise.
      107B      3203
      51      58      D0      107B      3204      $GETFLD      obi,l,num      ; Get the object number
      1088      3205      MOVL      R8,R1      ; Save object number
      1088      3206      $GETFLD      obi,s,fid      ; Setup field id
      10 50      E8      1098      3207      BLBS      R0,10$      ; If LBS then field is non-null
      109B      3208      $GETFLD      obi,s,nam      ; Else use the object's name
      58 50      E9      10A8      3209      BLBC      R0,30$      ; If LBC then null, filename is illegal
      51      D5      10AB      3210      10$:      TSTL      R1      ; Is this for object number 0?
      12      12      10AD      3211      BNEQ      20$      ; If NEQ no, use system defaults
      24      68      91      10AF      3212      CMPB      (R8),#^A'$'      ; Does name start with '$'?
      09      13      10B2      3213      BEQL      15$      ; If so, use system defaults
      63      68      57      28      10B4      3214      MOVCS      R7,(R8),(R3)      ; Else allow LOGIN to use user's defaults
      50      01      90      10B8      3215      MOVBS      #1,R0      ; Indicate success
      46      11      10BB      3216      BRB      30$      ; Continue
      58      D6      10BD      3217      15$:      INCL      R8      ; Strip off '$'
      57      D7      10BF      3218      DECL      R7
      50      000000E0'EF      9E      10C1      3219      20$:      MOVAB      NET$T_SYSFAB,R0      ; Setup for 'non-zero object' defaults
      34 A0      57      90      10C8      3220      MOVBS      R7,FAB$B_FNS(R0)      ; Set the current filename size
      2C A0      58      D0      10CC      3221      MOVL      R8,FAB$L_FNA(R0)      ; Set the current filename ptr
```


52	00000080'EF	9E	10D0	3222	MOVAB	NET\$T_PR\$NAM,R2	; Get output descriptor address
	0C A2 63	9E	10D7	3223	MOVAB	(R3),NAM\$\$_E\$A(R2)	; Set the buf ptr to rcv parse
			10DB	3224	\$PARSE	FAB = R0	; Get the filename
	52 0B A2	9A	10E4	3225	MOVZBL	NAM\$\$_E\$S(R2),R2	; Get the size of the filename
			10E8	3226			
			10E8	3227			; If the source string did not contain a trailing semicolon,
			10E8	3228			; strip it from the parsed string so that the image activator
			10E8	3229			; will use the installed version of the image (if any).
			10E8	3230			
	03	BB	10E8	3231	PUSHR	#^M<R0,R1>	; Preserve volatile registers.
68	57	3B	3A	10EA	LOCC	#^A''',R7,(R8)	; Find address of trailing '';
	0E	12	10EE	3232	BNEQ	25\$	; If NEQ, found it
63	52	3B	3A	10F0	LOCC	#^A''',R2,(R3)	; Find version number in parsed string
	52	50	C2	10F4	SUBL2	R0,R2	; Reduce size of string by size of ver.
0000008B'EF	52	90	10F7	3235	MOVAB	R2,NET\$T_PR\$NAM+NAM\$\$_E\$S	; ...also in the \$NAM block
	03	BA	10FE	3236	POPR	#^M<R0,R1>	; Restore volatile registers
	53	C0	1100	3237	ADDL	R2,R3	; Advance buffer pointer
		05	1103	3238	RSB		; Return status in R0
				3239			

```
1104 3241 .SBTTL ESI PARAMETER ACTION ROUTINES
1104 3242 :+
1104 3243 : NET$ESI_V_LCK - Get status of conditionally writeable fields
1104 3244 : NET$ESI_S_COL - Get collating value
1104 3245 :
1104 3246 : INPUTS: R11 NDI CNR address
1104 3247 : R10 NDI CNF address
1104 3248 : R9 FLD i.d. of field being read
1104 3249 : R1 Scratch
1104 3250 : R0 Scratch
1104 3251 :
1104 3252 : OUTPUTS: R1 Address of field value or longword string descriptor
1104 3253 : R0 Low bit set if R1 is valid
1104 3254 : Low bit clear otherwise
1104 3255 :
1104 3256 : All other register values are preserved.
1104 3257 :-
1104 3258 :-
1104 3259
1104 3260 NET$ESI_V_LCK:: ; Get status of cond. writeable fields
1104 3261 :
1104 3262 : If the state is not 'OFF' then the CNF is writelocked
1104 3263 :
1104 3264 : $GETFLD esi,l,sta ; Fetch UCB field
1104 3265 : BLBC R0,10$ ; If field isn't set then CNF not locked
1104 3266 : CMPL S^#NMASC_STATE_OFF,R8 ; Is the state 'OFF'
1104 3267 : BNEQ 10$ ; If not with R0=1
1104 3268 : CLRL R0 ; Otherwise, CNF is not locked
1104 3269 10$: MOVL R0,R1 ; Setup field value
1104 3270 : MOVL S^#SS$_NORMAL,R0 ; Success
1104 3271 : RSB ; Return to co-routine
1104 3272 :
1104 3273 NET$ESI_S_COL:: ; Get collating value
1104 3274 : $CNFFLD esi,l,snk,R9 ; Specify sink type
1104 3275 : BRB CONVERT ; Store it as a string
1104 3276 :
```

07 50 E9 1111 3265 \$GETFLD esi,l,sta ; Fetch UCB field
58 01 D1 1114 3266 BLBC R0,10\$; If field isn't set then CNF not locked
02 12 1117 3267 CMPL S^#NMASC_STATE_OFF,R8 ; Is the state 'OFF'
50 D4 1119 3268 BNEQ 10\$; If not with R0=1
51 50 D0 1118 3269 10\$: MOVL R0,R1 ; Otherwise, CNF is not locked
50 00' D0 111E 3270 : MOVL S^#SS\$_NORMAL,R0 ; Setup field value
05 1121 3271 : RSB ; Success
1122 3272 : ; Return to co-routine
1122 3273 NET\$ESI_S_COL:: ; Get collating value
1122 3274 : \$CNFFLD esi,l,snk,R9 ; Specify sink type
0D 11 1129 3275 : BRB CONVERT ; Store it as a string
112B 3276 :

```
112B 3278 .SBTTL EFI PARAMETER ACTION ROUTINES
112B 3279 :+
112B 3280 : NET$EFI_V_LCK - Get status of conditionally writeable fields
112B 3281 : NET$EFI_S_COL - Get collating value
112B 3282 :
112B 3283 : INPUTS: R11 NDI CNR address
112B 3284 : R10 NDI CNF address
112B 3285 : R9 FLD i.d. of field being read
112B 3286 : R1 Scratch
112B 3287 : R0 Scratch
112B 3288 :
112B 3289 : OUTPUTS: R1 Address of field value or longword string descriptor
112B 3290 : R0 Low bit set if R1 is valid
112B 3291 : Low bit clear otherwise
112B 3292 :
112B 3293 : All other register values are preserved.
112B 3294 :
112B 3295 :-
112B 3296 :
112B 3297 NET$EFI_V_LCK:: : Get status of cond. writeable fields
50 50 D4 112B 3298 C[RL R0 : CNF is never locked
01 01 D0 112D 3299 MOVL #1,R0 : Success
05 05 1130 3300 RSB : Return
1131 3301 :
1131 3302 NET$EFI_S_COL:: : Get collating value
1131 3303 $CNFFLD efi,l,sin,R9 : Specify sink node address
EECS' 30 1138 3304 CONVERT:BSBW CNF$GET_FIELD : Get its value
OE 50 E9 113B 3305 BLBC R0,10$ : If LBC then not active
58 DD 113E 3306 PUSHL R8 : Push it onto the stack
83 8E 90 1140 3307 MOVB (SP)+,(R3)+ : Move all 4 bytes to buffer, high
83 8E 90 1143 3308 MOVB (SP)+,(R3)+ : order byte first
83 8E 90 1146 3309 MOVB (SP)+,(R3)+
83 8E 90 1149 3310 MOVB (SP)+,(R3)+
05 114C 3311 10$: RSB : Retrun to co-routine
```

```
114D 3313 .SBTTL LLI PARAMETER ACTION ROUTINES
114D 3314 :+
114D 3315 : NET$LLI_V_LCK - See if CNF is write-locked
114D 3316 :
114D 3317 : NET$LLI_L_PID - Get process I.D. (external format)
114D 3318 : NET$LLI_L_IPID - Get process internal I.D.
114D 3319 : NET$LLI_L_DLY - Get round trip delay time
114D 3320 : NET$LLI_L_STA - Get link state
114D 3321 : NET$LLI_L_RLN - Get remote link number
114D 3322 : NET$LLI_L_LLN - Get local link number
114D 3323 : NET$LLI_L_PNA - Get remote node address from XWB
114D 3324 :
114D 3325 : NET$LLI_S_PNN - Get Partner node name
114D 3326 : NET$LLI_S_COL - Get collating value
114D 3327 : NET$LLI_S_PRC - Get owner process name
114D 3328 : NET$LLI_S_USR - Get owner user name
114D 3329 : NET$LLI_S_RID - Get remote user i.d.
114D 3330 : NET$LLI_S_CNT - Get link counters
114D 3331 :
114D 3332 : INPUTS: R11 LLI CNR address
114D 3333 : R10 LLI CNF address
114D 3334 : R9 FLD i.d. of field being read
114D 3335 : R1 Scratch
114D 3336 : R0 Scratch
114D 3337 :
114D 3338 : OUTPUTS: R1 Address of field value or longword string descriptor
114D 3339 : R0 Low bit set if R1 is valid
114D 3340 : Low bit clear otherwise
114D 3341 :
114D 3342 : R2-R5 may be destroyed.
114D 3343 :
114D 3344 : All other register values are preserved.
114D 3345 :
114D 3346 :-
114D 3347 :
114D 3348 NET$LLI_V_LCK:: : See if CNF is write-locked
114D 3349 ::88 MOVZBL #1,R1 : Field value
114D 3350 ::88 MOVZBL #SS$_NORMAL,R0 : Indicate success
114D 3351 ::88 RSB : Return
114D 3352 :
114D 3353 : CLRL R1 : Assume LLI is locked
114D 3354 : $GETFLD LLI,L_STA : Get the current STATE
114D 3355 : BLBC R0,30$ : Br if error in obtaining field
114D 3356 : CMPL #XWB$_STA_RUN,R8 : Is this link running?
114D 3357 : BEQL 30$ : Br if yes, LLI is locked
114D 3358 : MOVZBL #1,R1 : Else, LLI is unlocked
114D 3359 30$: MOVZBL #SS$_NORMAL,R0 : Indicate success
114D 3360 : RSB
114D 3361 :
114D 3362 NET$LLI_L_PID:: : Get external PID
114D 3363 : MOVL CNF$_C_LENGTH - : Get address of XWB
114D 3364 : +LLI$_XWB(R10),R0
114D 3365 : MOVL XWB$_PID(R0),R0 : Get PID
114D 3366 : JSB G^EXE$IPID_TO_EPID : Convert it
114D 3367 : MOVL R0,R1 : Return it
114D 3368 : MOVZBL #SS$_NORMAL,R0 : Indicate success
114D 3369 : RSB : Return

51 D4 114D 3353 CLRL R1 : Assume LLI is locked
08 50 E9 114F 3354 $GETFLD LLI,L_STA : Get the current STATE
58 05 D1 115C 3355 BLBC R0,30$ : Br if error in obtaining field
03 13 1162 3356 CMPL #XWB$_STA_RUN,R8 : Is this link running?
51 01 9A 1164 3357 BEQL 30$ : Br if yes, LLI is locked
50 00'8F 9A 1167 3358 MOVZBL #1,R1 : Else, LLI is unlocked
05 1168 3359 30$: MOVZBL #SS$_NORMAL,R0 : Indicate success
116C 3360 : RSB
116C 3361 :
50 24 AA D0 116C 3362 NET$LLI_L_PID:: : Get external PID
00000000'GF 16 1170 3363 : MOVL CNF$_C_LENGTH - : Get address of XWB
51 50 D0 1170 3364 : +LLI$_XWB(R10),R0
50 00'8F 9A 1174 3365 : MOVL XWB$_PID(R0),R0 : Get PID
05 117A 3366 : JSB G^EXE$IPID_TO_EPID : Convert it
117D 3367 : MOVL R0,R1 : Return it
1181 3368 : MOVZBL #SS$_NORMAL,R0 : Indicate success
3369 : RSB : Return
```

```
1182 3370
1182 3371 NET$LLI-L IPID:: ; Get Internal PID
50 24 AA D0 1182 3372 ; Get address of XWB
1186 3373
1186 3374 MOVL CNF$C_LENGTH -
51 34 A0 D0 1186 3375 +LLIS[ XWB(R10),R0
50 00'8F 9A 118A 3376 XWB$P_PID(R0),R1 ; Return PID
05 118E 3377 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
118F 3378 RSB ; Return
118F 3379
118F 3378 NET$LLI-L DLY:: ; Get address of XWB
50 24 AA D0 118F 3379 +LLIS[ XWB(R10),R0
1193 3380 ; Return the round trip delay
51 4E A0 3C 1193 3381 MOVZWL XWB$W_DELAY(R0),R1
50 00'8F 9A 1197 3382 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 119B 3383 RSB ; Return
119C 3384
119C 3385 NET$LLI-L RLN:: ; Get address of XWB
50 24 AA D0 119C 3386 +LLIS[ XWB(R10),R0
11A0 3387 ; Return the remote link number
51 3C A0 3C 11A0 3388 MOVZWL XWB$W_REMLNK(R0),R1
50 00'8F 9A 11A4 3389 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11A8 3390 RSB ; Return
11A9 3391
11A9 3392 NET$LLI-L LLN:: ; Get address of XWB
50 24 AA D0 11A9 3393 +LLIS[ XWB(R10),R0
11AD 3394 ; Return the remote link number
51 3E A0 3C 11AD 3395 MOVZWL XWB$W_OCLNK(R0),R1
50 00'8F 9A 11B1 3396 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11B5 3397 RSB ; Return
11B6 3398
11B6 3399 NET$LLI-L PNA:: ; Get address of XWB
50 24 AA D0 11B6 3400 +LLIS[ XWB(R10),R0
11BA 3401 ; Return the remote link number
51 3A A0 3C 11BA 3402 MOVZWL XWB$W_REMNOD(R0),R1
50 00'8F 9A 11BE 3403 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11C2 3404 RSB ; Return
11C3 3405
11C3 3406 NET$LLI-L STA:: ; Get address of XWB
50 24 AA D0 11C3 3407 +LLIS[ XWB(R10),R0
11C7 3408 ; Return the link state
51 1E A0 9A 11C7 3409 MOVZBL XWB$B_STA(R0),R1
50 00'8F 9A 11CB 3410 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11CF 3411 RSB ; Return
11D0 3412
11D0 3413 NET$LLI-S CNT:: ; Get link counters
50 00'8F 9A 11D0 3414 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11D4 3415 RSB ; Return
11D5 3416
11D5 3417 NET$LLI-S RID:: ; Get remote user i.d.
50 24 AA D0 11D5 3418 MOVL CNF$C_LENGTH -
11D9 3419 +LLIS[ XWB(R10),R0 ; Get address of XWB
51 6F A0 9A 11D9 3420 MOVZBL XWB$B_RID(R0),R1 ; Get remote user i.d. string length
09 13 11DD 3421 BEQL 10$ ; If EQL return with LBC in R0
63 70 A0 51 28 11DF 3422 MOVCL R1,XWB$T_RID(R0),(R3) ; Move the name
50 00'8F 9A 11E4 3423 MOVZBL #SS$ _NORMAL,R0 ; Indicate success
05 11E8 3424 10$: RSB ; Return
11E9 3425
11E9 3426
```

```

50 24 AA D0 11E9 3427 NET$LLI_S_PRC::          : Get owner process name
                                11E9 3428 -MOVL      CNF$C_LENGTH -
                                11ED 3429 +LLIS[ XWB(R'0),R0
50 34 A0 D0 11ED 3430      MOVL      XWB$P_PID(R0),R0      : Get PID
                                11F1 3431 BSBB      GET_JPI      : Get Job/process info
                                11F3 3432 BLBC      R0,TOS      : If LBC then info not found
50 00000050'EF 9A 11F6 3433 MOVZBL    PNAME$,R0      : Get string size
                                11FD 3434 BEQL      10$      : If EQL return with LBC in R0
63 00000054'EF 50 28 11FF 3435 MOVCL   R0,PNAME,(R3)      : Move the process name
50 00'8F 9A 1207 3436 MOVZBL    #$$$_NORMAL,R0      : Indicate success
                                05 1208 3437 10$: RSB      : Return
                                120C 3438
                                120C 3439 NET$LLI_S_USR::          : Get owner user name
50 24 AA D0 120C 3440 -MOVL      CNF$C_LENGTH -
                                1210 3441 +LLIS[ XWB(R10),R0
50 34 A0 D0 1210 3442      MOVL      XWB$P_PID(R0),R0      : Get PID
                                1214 3443 BSBB      GET_JPI      : Get Job/process info
                                1216 3444 BLBC      R0,TOS      : If LBC then info not found
50 00000040'EF 9A 1219 3445 MOVZBL    UNAME$,R0      : Get string size
                                1220 3446 BEQL      10$      : If EQL return with LBC in R0
63 00000044'EF 50 28 1222 3447 MOVCL   R0,UNAME,(R3)      : Move the username
50 00'8F 9A 122A 3448 MOVZBL    #$$$_NORMAL,R0      : Indicate success
                                05 122E 3449 10$: RSB      : Return
                                122F 3450
                                122F 3451 GET_JPI:          : Get Job/process info
50 D5 122F 3452 TSTL      R0      : Any PID yet?
3C 13 1231 3453 BEQL      10$      : If EQL no, return LBC in R0
00000000'GF 16 1233 3454 JSB      G^EXES$IPID_TO_EPID      : Convert to EPID
50 DD 1239 3455 PUSHL     R0      : Save EPID on stack for call
50 5E D0 123B 3456 MOVL     SP,R0      : Get address of EPID
                                123E 3457 $GETJPI S -
                                123E 3458 PIDADR = (R0),-      : EPID of process of interest
                                123E 3459 EFN = #NET$C_EFN_WAIT,- : Event flag
                                123E 3460 IOSB = IOSB,-      : IOSB
                                123E 3461 ITMLST = ITEM_LIST      : Item list for return
50 SE 04 C0 1259 3462 ADDL      #4,SP      : Pop EPID off stack
10 50 E9 125C 3463 BLBC      R0,10$      : Br on error
50 00000038'EF 3C 125F 3464 $WAITFR S EFN = #NET$C_EFN_WAIT : Wait for $GETJPI to finish
05 1268 3465 MOVZWL -IOSB,R0      : Setup status
                                126F 3466 10$: RSB      : Return status in R0
                                1270 3467
                                1270 3468 NET$LLI_S_COL::          : Collating value
50 24 AA D0 1270 3469 -MOVL      CNF$C_LENGTH -
                                1274 3470 +LLIS[ XWB(R10),R0
83 3B A0 90 1274 3471      MOVBL     XWB$W_REMNOD+1(R0),(R3)+ : Insert remnde node address
                                1278 3472      : ... high order first
83 3A A0 90 1278 3473      MOVBL     XWB$W_REMNOD(R0),(R3)+
50 3E A0 60 127C 3474      MOVBL     XWB$W_LOCLNK(R0),R0      : Get logical link number
7E 50 FC00 8F AB 1280 3475 BICW3     #^C<NET$M_MAXLNKMSK>,R0,-(SP) : Use index bits only
8E 95 1286 3476 TSTB      (SP)+      : Pop low order bits (they're in R0)
83 8E 90 1288 3477      MOVBL     (SP)+,(R3)+      : Insert high order
83 50 90 1288 3478      MOVBL     R0,(R3)+      : Insert low order
50 00'8F 9A 128E 3479 MOVZBL    #$$$_NORMAL,R0      : Indicate success
                                05 1292 3480 RSB
                                1293 3481
                                1293 3482 NET$LLI_S_PNN::          : Partner node name
5B 00000000'EF D0 1293 3483 -MOVL     NET$GL_CNR_NDI,R11      : Get the NDI root block
```

50	24	AA	D0	129A	3484	MOVL	CNF\$C_LENGTH -	; Get address of XWB
				129E	3485		+LLIS[XWB(R10),R0	
58	3A	A0	3C	129E	3486	MOVZWL	XWB\$W_REMNOD(R0),R8	; Get node address
		0280	30	12A2	3487	BSBW	NET\$NDI_BY_ADD	; Get associated NDI CNF address
		18 50	E9	12A5	3488	BLBC	R0,10\$	; If LBC then no NDI
				12A8	3489	\$GETFLD	ndi,s,nna	; Get node name
	08	50	E9	12B5	3490	BLBC	R0,10\$	; Branch if not present
63	68	57	28	12B8	3491	MOVC	R7,(R8),(R3)	; Copy into buffer
50	00	8F	9A	12BC	3492	MOVZBL	#SS\$_NORMAL,R0	; Indicate successful
			05	12C0	3493	RSB		; Return status in R0

```
12C1 3495 .SBTTL SPI PARAMETER ACTION ROUTINES
12C1 3496 :+
12C1 3497 : NET$SPI_S_COL - Get collating value
12C1 3498 :
12C1 3499 : INPUTS: R11 CNR address
12C1 3500 : R10 CNF address
12C1 3501 : R9 FLD i.d. of field being read
12C1 3502 : R3 Address of result buffer
12C1 3503 :
12C1 3504 : OUTPUTS: R1 Address of field value or longword string descriptor
12C1 3505 : R0 Low bit set if R1 is valid
12C1 3506 : Low bit clear otherwise
12C1 3507 :
12C1 3508 : All other register values are preserved.
12C1 3509 :-
12C1 3510
12C1 3511 NET$SPI_V_LCK:: : 'CNF locked' flag
50 01 D0 12C1 3512 : MOVL #1,R0 : Mark all 'cond write' fields as
05 12C4 3513 : RSB : cannot be written.
12C5 3514
12C5 3515 NET$SPI_S_COL:: : Get collating value
12C5 3516 : $GETFLD spi,l,pid : Get server process PID
09 50 E9 12D2 3517 : BLBC R0,90$ : Branch if not set
7E 58 B0 12D5 3518 : MOVW R8,-(SP) : Push low order word (process index)
83 8E 90 12D8 3519 : MOVB (SP)+,(R3)+ : Invert bytes, move to buffer
83 8E 90 12DB 3520 : MOVB (SP)+,(R3)+
05 12DE 3521 90$: RSB
```



```
12DF 3523 .SBTTL AJI PARAMETER ACTION ROUTINES
12DF 3524 ;+
12DF 3525 : NETSAJI_V_REA - Get adjacency 'run' status
12DF 3526 :
12DF 3527 : NETSAJI_L_ADD - Get partner node address
12DF 3528 : NETSAJI_L_TYP - Get partner node type
12DF 3529 : NETSAJI_L_LIT - Get adjacency listen interval
12DF 3530 : NETSAJI_L_BLO - Get partner block size
12DF 3531 : NETSAJI_L_RPR - Get partner broadcast router priority
12DF 3532 :
12DF 3533 : NETSAJI_S_COL - Get collating value
12DF 3534 : NETSAJI_S_NNA - Get partner node name
12DF 3535 : NETSAJI_S_CIR - Get circuit name
12DF 3536 :
12DF 3537 : INPUTS: R11 CNR address
12DF 3538 : R10 CNF address
12DF 3539 : R9 FLD i.d. of field being read
12DF 3540 : R3 Address of result buffer
12DF 3541 :
12DF 3542 : OUTPUTS: R1 Address of field value or longword string descriptor
12DF 3543 : R0 Low bit set if R1 is valid
12DF 3544 : Low bit clear otherwise
12DF 3545 :
12DF 3546 : All other register values are preserved.
12DF 3547 :-
12DF 3548
12DF 3549 NETSAJI_V_LCK:: ; 'CNF locked' flag
50 01 D0 12DF 3550 -MOVL #1,R0 ; Mark all 'cond write' fields as
05 12E2 3551 RSB ; cannot be written.
12E3 3552
12E3 3553 NETSAJI_V_REA::
00A6 30 12E3 3554 BSBW LOCATE_ADJ ; Lookup adjacency
05 50 E9 12E6 3555 BLBC RO,90$ ; Branch if error
51 67 01 01 EF 12E9 3556 EXTZV #ADJSV_RUN,#1,ADJSB_STS(R7),R1 ; Get flag
05 12EE 3557 90$: RSB
12EF 3558
12EF 3559 NETSAJI_L_ADD::
009A 30 12EF 3560 BSBW LOCATE_ADJ ; Lookup adjacency
0C 50 E9 12F2 3561 BLBC RO,90$ ; Branch if error
51 04 A7 3C 12F5 3562 MOVZWL ADJSW_PNA(R7),R1 ; Return node address
04 13 12F9 3563 BEQL 80$ ; If none, then return 'not set'
FC3A 30 12FB 3564 BSBW SUPPRESS_AREA ; Suppress area if necessary
05 12FE 3565 RSB
12FF 3566
12FF 3567 80$: CLRL R0 ; Indicate parameter 'not set'
05 1301 3568 90$: RSB
1302 3569
1302 3570 NETSAJI_L_TYP::
0087 30 1302 3571 BSBW LOCATE_ADJ ; Lookup adjacency
04 50 E9 1305 3572 BLBC RO,90$ ; Branch if error
51 01 A7 9A 1308 3573 MOVZBL ADJSB_PTYPE(R7),R1 ; Return node type
05 130C 3574 90$: RSB
130D 3575
130D 3576 NETSAJI_L_LIT::
007C 30 130D 3577 BSBW LOCATE_ADJ ; Lookup adjacency
0C 50 E9 1310 3578 BLBC RO,90$ ; Branch if error
06 67 01 E1 1313 3579 BBC #ADJSV_RUN,ADJSB_STS(R7),80$ ; Branch if ADJ not up
```

```
51 08 A7 3C 1317 3580 MOVZWL ADJ$W_INT_LSN(R7),R1 ; Return listen interval
      02 12 131B 3581 BNEQ 90$ ; Ok if non-zero
      50 D4 131D 3582 80$: CLRL R0 ; Indicate parameter 'not set'
      05 131F 3583 90$: RSB
      1320 3584
      1320 3585 NET$AJI_L_BLO::
      0069 30 1320 3586 BSBW LOCATE_ADJ ; Lookup adjacency
      0C 50 E9 1323 3587 BLBC R0,90$ ; Branch if error
      06 67 01 E1 1326 3588 BBC #ADJ$V_RUN,ADJ$B_STS(R7),80$ ; Branch if ADJ not up
51 06 A7 3C 132A 3589 MOVZWL ADJ$W_BUFSIZ(R7),R1 ; Return partner block size
      02 12 132E 3590 BNEQ 90$ ; Ok if non-zero
      50 D4 1330 3591 80$: CLRL R0 ; Indicate parameter 'not set'
      05 1332 3592 90$: RSB
      1333 3593
      1333 3594 NET$AJI_L_RPR::
      0056 30 1333 3595 BSBW LOCATE_ADJ ; Lookup adjacency
      0D 50 E9 1336 3596 BLBC R0,90$ ; Branch if error
      50 D4 1339 3597 CLRL R0 ; Assume failure
      7 67 01 E1 133B 3598 BBC #ADJ$V_RUN,ADJ$B_STS(R7),90$ ; Branch if ADJ not up
51 0C A7 9A 133F 3599 MOVZBL ADJ$B_BCPR1(R7),R1 ; Return broadcast router priority
      50 00 D0 1343 3600 MOVL S^#SS$_NORMAL,R0 ; Successful
      05 1346 3601 90$: RSB
      1347 3602
      1347 3603 NET$AJI_S_COL::
      7E 12 AA B0 1347 3604 MOVW CNF$W_ID(R10),-(SP) ; Get collating value
      83 8E 90 134B 3605 MOVW (SP)+,(R3)+ ; Push ADJ index
      83 8E 90 134E 3606 MOVW (SP)+,(R3)+ ; Invert bytes, move to buffer
      50 00 D0 1351 3607 MOVL S^#SS$_NORMAL,R0 ; Successful
      05 1354 3608 RSB
      1355 3609
      1355 3610 NET$AJI_S_NNA::
      35 10 1355 3611 BSBW LOCATE_ADJ ; Lookup adjacency
      21 50 E9 1357 3612 BLBC R0,90$ ; Branch if error
      50 D4 135A 3613 CLRL R0 ; Assume failure
      1B 67 01 E1 135C 3614 BBC #ADJ$V_RUN,ADJ$B_STS(R7),90$ ; Branch if ADJ not up
58 04 A7 3C 1360 3615 MOVZWL ADJ$W_PNA(R7),R8 ; Get partner node address
58 00000000 EF D0 1364 3616 MOVL NET$GC_CNR_NDI,R11 ; Get NDI root address
      01B7 30 136B 3617 BSBW NET$NDI_BY_ADD ; Locate NDI CNF block
      0A 50 E9 136E 3618 BLBC R0,90$ ; Branch if not found
      0255 30 1371 3619 $CNFFLD ndi,s,nna,R9 ; Set node name field ID
      05 1378 3620 BSBW MOVSTR ; Copy it to output buffer
      137B 3621 90$: RSB
      137C 3622
      137C 3623 NET$AJI_S_CIR::
      0E 10 137C 3624 BSBW LOCATE_ADJ ; Lookup adjacency
      0A 50 E9 137E 3625 BLBC R0,90$ ; Branch if error
      1381 3626 $CNFFLD cri,s,nam,R9 ; Set circuit name field ID
      0245 30 1388 3627 BSBW MOVSTR ; Copy it to output buffer
      05 138B 3628 90$: RSB
      138C 3629
      138C 3630 LOCATE_ADJ:
      58 12 AA 3C 138C 3631 MOVZWL CNF$W_ID(R10),R8 ; Get ADJ index
      EC6D 30 1390 3632 BSBW NET$ADJ_LPD_CRI ; Get CRI CNF, LPD & ADJ pointers
      05 1393 3633 RSB
```

```
1394 3635 .SBTTL SDI PARAMETER ACTION ROUTINES
1394 3636 :+
1394 3637 : NET$SDI_L_SUB - Get DLE substate
1394 3638 : NET$SDI_L_PID - Get PID of process owning DLE link
1394 3639 :
1394 3640 : NET$SDI_S_COL - Get collating value
1394 3641 : NET$SDI_S_CIR - Get circuit for DLE link
1394 3642 : NET$SDI_S_PHA - Get DLE physical address (BC only)
1394 3643 : NET$SDI_S_PRC - Get name of process owning DLE link
1394 3644 :
1394 3645 : INPUTS: R11 CNR address
1394 3646 : R10 CNF address
1394 3647 : R9 FLD i.d. of field being read
1394 3648 : R_ Address of result buffer
1394 3649 :
1394 3650 : OUTPUTS: R1 Address of field value or longword string descriptor
1394 3651 : R0 Low bit set if R1 is valid
1394 3652 : Low bit clear otherwise
1394 3653 :
1394 3654 : All other register values are preserved.
1394 3655 :-
1394 3656 :
1394 3657 NET$SDI_V_LCK:: : 'CNF locked' flag
50 01 D0 1394 3658 : MOVL #1,R0 ; Mark all 'cond write' fields as
05 1397 3659 : RSB ; cannot be written.
1398 3660 :
1398 3661 GET_DWB_ADR:
56 26 AA D0 1398 3662 : MOVL CNF$C_LENGTH+2(R10),R6 ; Get saved DWB address from scan routine
05 139C 3663 : RSB
139D 3664 :
139D 3665 NET$SDI_L_SUB::
51 46 F9 10 139D 3666 : GET DWB ADR ; Get DWB address
9A 139F 3667 : MOVZBL DWB$B_SOBSTA(R6),R1 ; Return value
50 01 D0 13A3 3668 : MOVL #1,R0 ; Success
05 13A6 3669 : RSB
13A7 3670 :
13A7 3671 NET$SDI_L_PID::
51 34 EF 10 13A7 3672 : GET DWB ADR ; Get DWB address
D0 13A9 3673 : MOVL DWB$L_PID(R6),R1 ; Return value
50 01 D0 13AD 3674 : MOVL #1,R0 ; Success
05 13B0 3675 : RSB
13B1 3676 :
13B1 3677 NET$SDI_S_COL::
7E 12 AA B0 13B1 3678 : MOVW CNF$W_ID(R10),-(SP) ; Push DWB identifier
83 8E 90 13B5 3679 : MOVW (SP)+,(R3)+ ; Copy reversing all the bytes
83 8E 90 13B8 3680 : MOVW (SP)+,(R3)+
50 01 D0 13BB 3681 : MOVL #1,R0 ; Success
05 13BE 3682 : RSB
13BF 3683 :
13BF 3684 NET$SDI_S_CIR::
58 07 D7 10 13BF 3685 : GET DWB ADR ; Get DWB address
3E A6 3C 13C1 3686 : MOVZWL DWB$W_PATH(R6),R8 ; Get LPD ID
EC 38 30 13C5 3687 : BSBW NET$GET_LPD_CRI ; Get CRI CNF, LPD pointers
0A 50 E9 13C8 3688 : BLBC R0,90$ ; Branch if error
01FB 30 13CB 3689 : $CNFFLD cri,s,nam,R9 ; Set circuit name field ID
05 13D2 3690 : BSBW MOVSTR ; Copy it to output buffer
13D5 3691 90$: RSB
```

			10	13D6	3692			
			D0	13D6	3693	NET\$SDI_S_PHA::		
83	40	C0	D0	13D6	3694	BSBB	GET_DWB_ADR	; Get DWB address
83	44	A6	B0	13D8	3695	MOVL	DWB\$G_REMNOD(R6),(R3)+	; Copy into buffer
	50	01	D0	13DC	3696	MOVW	DWB\$G_REMNOD+4(R6),(R3)+	
			D0	13E0	3697	MOVL	#1,R0	; Success
			05	13E3	3698	RSB		
				13E4	3699			
				13E4	3700	NET\$SDI_S_PRC::		
		B2	10	13E4	3701	BSBB	GET_DWB_ADR	; Get DWB address
50	34	A6	D0	13E6	3702	MOVL	DWB\$P_PID(R6),R0	; Pass process PID
		FE42	30	13EA	3703	BSBW	GET_JPI	; Get process name
50	00000050	'EF	9A	13ED	3704	MOVZBL	PNAME,R0	; Get string size
		0B	13	13F4	3705	BEQL	90\$	
63	00000054	'EF	28	13F6	3706	MOVC3	R0,PNAME,(R3)	; Copy process name into buffer
	50	01	D0	13FE	3707	MOVL	#1,R0	; Success
			05	1401	3708	RSB		

```
1402 3710 .SBTTL ARI PARAMETER ACTION ROUTINES
1402 3711 :+
1402 3712 : NETSARI_V_REA - Get adjacency "run" status
1402 3713 :
1402 3714 : NETSARI_L_ADD - Get partner node address
1402 3715 : NETSARI_L_DCO - Get cost to area
1402 3716 : NETSARI_L_DHO - Get hops to area
1402 3717 : NETSARI_L_NND - Get next node to area
1402 3718 :
1402 3719 : NETSARI_S_COL - Get collating value
1402 3720 : NETSARI_S_CIR - Get circuit name
1402 3721 :
1402 3722 : INPUTS: R11 CNR address
1402 3723 : R10 CNF address
1402 3724 : R9 FLD i.d. of field being read
1402 3725 : R3 Address of result buffer
1402 3726 :
1402 3727 : OUTPUTS: R1 Address of field value or longword string descriptor
1402 3728 : R0 Low bit set if R1 is valid
1402 3729 : Low bit clear otherwise
1402 3730 :
1402 3731 : All other register values are preserved.
1402 3732 :-
1402 3733 :
1402 3734 NETSARI_V_LCK:: : 'CNF locked' flag
50 01 D0 1402 3735 MOVL #1,R0 : Mark all "cond write" fields as
05 1405 3736 RSB : cannot be written.
1406 3737 :
1406 3738 NETSARI_V_REA:: : Area reachability
52 12 AA 3C 1406 3739 MOVZWL CNFSW_ID(R10),R2 : Get area number
00A4 30 140A 3740 BSBW NET$AREA_REACH : Determine if area reachable
51 50 9A 140D 3741 MOVZBL R0,R1 : Return boolean true/false
50 00' D0 1410 3742 MOVL S^#SS$_NORMAL,R0 : Return successful
05 1413 3743 RSB
1414 3744 :
1414 3745 NETSARI_L_ADD:: : Area address
51 12 AA 3C 1414 3746 MOVZWL CNFSW_ID(R10),R1 : Return area number
50 00' D0 1418 3747 MOVL S^#SS$_NORMAL,R0 : Return successful
05 141B 3748 RSB
141C 3749 :
141C 3750 NETSARI_L_DCO:: : Cost to area
51 51 0A 1A 10 141C 3751 BSBW AREA_COST_HOPS : Get cost/hops value
08 50 E9 141E 3752 BLBC R0,90$ : Exit if don't know
0A 00 EF 1421 3753 EXTZV #0,#10,R1,R1 : Get cost
50 00' D0 1426 3754 MOVL S^#SS$_NORMAL,R0 : Successful
05 1429 3755 90$: RSB
142A 3756 :
142A 3757 NETSARI_L_DHO:: : Hops to area
51 51 05 0C 10 142A 3758 BSBW AREA_COST_HOPS : Get cost/hops value
08 50 E9 142C 3759 BLBC R0,90$ : Exit if don't know
0A 0A EF 142F 3760 EXTZV #10,#5,R1,R1 : Get hops
50 00' D0 1434 3761 MOVL S^#SS$_NORMAL,R0 : Successful
05 1437 3762 90$: RSB
1438 3763 :
1438 3764 AREA_COST_HOPS: : Assume success
51 50 00' 9A 1438 3765 MOVZBL S^#SS$_NORMAL,R0 : Get area number
12 AA 3C 143B 3766 MOVZWL CNFSW_ID(R10),R1
```

```
57 00000000'EF D0 143F 3767 MOVL NET$GL_PTR_VCB,R7 ; Get RCB address
    03 008A C7 91 1446 3768 CMPB RCB$B_ETY(R7),#ADJ$C_PTY_AREA ; Are we an area router?
      OE 12 144B 3769 BNEQ 10$ ; Branch if not
      00 E1 144D 3770 BBC #RCB$V_LVL2,- ; If we are not enabled for Level 2
    09 08 A7 144F 3771 RCB$B_STATUS(R7),10$ ; routing, then act like a Level 1 router
51 00000000'GF 3C 1452 3772 MOVZWL G^NET$AW_AREA_C_H[R1],R1 ; Get cost/hops for area
    05 145A 3773 RSB
    05 008A C7 91 145B 3774 10$: CMPB RCB$B_ETY(R7),#ADJ$C_PTY_PH4N ; Are we an endnode?
      08 13 1460 3775 BEQL 20$ ; Branch if so
51 00000000'GF 3C 1462 3776 MOVZWL G^NET$AW_MIN_C_H,R1 ; Get cost/hops to 'nearest level 2 router'
    05 1469 3777 RSB
    50 D4 146A 3778 20$: CLRL R0 ; Don't know cost/hops
    05 146C 3779 RSB
      146D 3780
      146D 3781 NET$ARI_L_NND:: ; Next node on way to area
52 12 AA 3C 146D 3782 MOVZWL CNF$W_ID(R10),R2 ; Get area number
      003D 30 1471 3783 BSBW NET$AREA_REACH ; Determine output ADJ to area
      10 50 E9 1474 3784 BLBC R0,90$ ; If not reachable, then failure
      58 51 3C 1477 3785 MOVZWL R1,R8 ; Set ADJ index
      EB83' 30 147A 3786 BSBW NET$FIND_ADJ ; Get next hops ADJ address
      07 50 E9 147D 3787 BLBC R0,90$ ; If not found, then failure
51 04 A7 3C 1480 3788 MOVZWL ADJ$W_PNA(R7),R1 ; Return partner's node address
      FAB1 30 1484 3789 BSBW SUPPRESS_AREA ; Suppress area if necessary
    05 1487 3790 90$: RSB
      1488 3791
      1488 3792 NET$ARI_S_COL:: ; Get collating value
7E 12 AA B0 1488 3793 MOVW CNF$W_ID(R10),-(SP) ; Push area number
      83 8E 90 148C 3794 MOVB (SP)+,(R3)+ ; Invert bytes, move to buffer
      83 8E 90 148F 3795 MOVB (SP)+,(R3)+
      50 00' D0 1492 3796 MOVL S^#SS$_NORMAL,R0 ; Successful
    05 1495 3797 RSB
      1496 3798
      1496 3799 NET$ARI_S_DLI:: ; Circuit used to get to area
52 12 AA 3C 1496 3800 MOVZWL CNF$W_ID(R10),R2 ; Get area number
      0014 30 149A 3801 BSBW NET$AREA_REACH ; Determine output ADJ to area
      58 51 3C 149D 3802 MOVZWL R1,R8 ; Set ADJ index to area
      EB5D' 30 14A0 3803 BSBW NET$ADJ_LPD_CRI ; Get CRI CNF, LPD & ADJ pointers
      0A 50 E9 14A3 3804 BLBC R0,90$ ; Branch if error detected
      0120 30 14A6 3805 $CNFFLD cri,s,nam,R9 ; Set circuit name field ID
    05 14AD 3806 BSBW MOVSTR ; Copy it to output buffer
      14B0 3807 90$: RSB
```

```
14B1 3809 .SBTTL NET$AREA_REACH - Test area reachability
14B1 3810 :+
14B1 3811 : NET$AREA_REACH - Test area reachability
14B1 3812 :
14B1 3813 : This routine tests the reachability of an area, and returns:
14B1 3814 :
14B1 3815 : 1) whether it is reachable or not
14B1 3816 : 2) the ADJ to get to the area
14B1 3817 :
14B1 3818 : Inputs: R2 Area address
14B1 3819 : R1 Scratch
14B1 3820 : R0 Scratch
14B1 3821 :
14B1 3822 : Outputs: R2 Area address
14B1 3823 : R1 Adjacency index of path used to reach the area
14B1 3824 : R0 Status
14B1 3825 :-
14B1 3826
14B1 3827 NET$AREA_REACH:: : Test for area reachability
51 50 0000'8F DD 14B1 3828 PUSHL #0 : Init storage on stack
00000000'EF DO 14B3 3829 MOVZWL #SS$ NOSUCHNODE,R0 : Assume area out of range
008C C1 52 91 14B8 3830 MOVL NET$GL_PTR_VCB,R1 : Get the RCB
2D 1A 14BF 3831 CMPB R2,RCB$B_MAX_AREA(R1) : Within range?
03 008A C1 91 14C4 3832 BGTRU 100$ : If GTRU then out of range
0C 12 14C6 3833 CMPB RCB$B_ETY(R1),#ADJ$C_PTY_AREA : Are we a level 2 router?
6E 20 B142 3C 14CB 3834 BNEQ 50$ : If not, use nearest level 2 router
1A 13 14CD 3835 MOVZWL @RCB$B_PTR_AOA(R1)[R2],(SP) : Get ADJ index to the area
50 00' DO 14D4 3836 BEQL 80$ : If 0, then unreachable
1A 11 14D7 3837 40$: MOVL S^#SS$_NORMAL,R0 : Indicate reachable
05 008A C1 91 14D9 3838 BRB 100$ : And exit with success
07 13 14DE 3839 50$: CMPB RCB$B_ETY(R1),#ADJ$C_PTY_PH4N : Are we an endnode?
6E 00AC C1 3C 14E0 3840 BEQL 60$ : Branch if so
ED 11 14E5 3841 MOVZWL RCB$W_LVL2(R1),(SP) : Get ADJ index to nearest level2 router
6E 00AA C1 3C 14E7 3842 BRB 40$ : and always exit with success
E6 12 14E9 3843 60$: MOVZWL RCB$W_DRT(R1),(SP) : Get ADJ index to designated router
50 0000'8F 3C 14EC 3844 BNEQ 40$ : and exit with success if we have one
51 8ED0 14F3 3845 80$: MOVZWL #SS$_UNREACHABLE,R0 : Node is known, but unreachable
05 14F6 3846 100$: POPL R1 : Return path in R1
RSB
```

```
14F7 3849 .SBTTL NET$GET_LOC_STA - GET EXECUTOR STATE
14F7 3850 :++
14F7 3851 : NET$GET_LOC_STA - Get the state of the local node
14F7 3852 :
14F7 3853 : INPUTS:      None
14F7 3854 :
14F7 3855 : OUTPUTS:    R0      State value
14F7 3856 :
14F7 3857 :          All other registers are preserved
14F7 3858 :
14F7 3859 :--
14F7 3860 NET$GET_LOC_STA::
14F7 3861     PUSHRR    #^M<R7,R8,R9,R10,R11>      ; Return local state in R0
14FB 3862     MOVL    NET$GL_CNR_LNI,R11          ; Save regs
1502 3863     MOVL    CNR$L_FLINK(R11),R10         ; Setup the Root block ptr
1505 3864     BBS     #CNF$V_FLG_CNR,-            ; Get the first CNF block
1507 3865             CNF$B_FLG(R10),10$         ; If its the root then the CNF list
150A 3866             $GETFLD lni,l_sta          ; is empty
1517 3867     BLBS     R0,20$                     ; Get the local state
151A 3868 10$:     MOVW    #LNISC_STA_OFF,R8       ; Br if valid
151D 3869 20$:     MOVL    R8,R0                  ; Assume the 'off' state
1520 3870     POPR     #^M<R7,R8,R9,R10,R11>     ; Get the state value
1524 3871     RSB                      ; Restore regs
```

SB 0F80 8F BB 14F7 3861 PUSHRR #^M<R7,R8,R9,R10,R11> ; Return local state in R0
 00000000'EF DO 14FB 3862 MOVL NET\$GL_CNR_LNI,R11 ; Save regs
 5A 6B DO 1502 3863 MOVL CNR\$L_FLINK(R11),R10 ; Setup the Root block ptr
 00 EO 1505 3864 BBS #CNF\$V_FLG_CNR,- ; Get the first CNF block
 10 0B AA 1507 3865 CNF\$B_FLG(R10),10\$; If its the root then the CNF list
 03 50 E8 150A 3866 \$GETFLD lni,l_sta ; is empty
 58 01 B0 1517 3867 BLBS R0,20\$; Get the local state
 50 58 DO 151A 3868 10\$: MOVW #LNISC_STA_OFF,R8 ; Br if valid
 0F80 8F BA 151D 3869 20\$: MOVL R8,R0 ; Assume the 'off' state
 1520 3870 POPR #^M<R7,R8,R9,R10,R11> ; Get the state value
 1524 3871 RSB ; Restore regs


```
1525 3873 .SBTTL NET$NDI_BY_ADD - Find NDI CNF by node address
1525 3874 :
1525 3875 : NET$NDI_BY_ADD - Find NDI CNF by node address
1525 3876 :
1525 3877 : FUNCTIONAL DESCRIPTION:
1525 3878 :
1525 3879 : The node address is used as an index into the NDI vector in order to locate
1525 3880 : corresponding CNF block. Only "real" NDI CNF blocks are considered valid,
1525 3881 : the so called "phantom" and "loop" node CNFs are not returned.
1525 3882 :
1525 3883 : This routine is merely an optimization. It could be replaced with a call
1525 3884 : to $SEARCH eql,ndi,l,add. The optimization is desirable since this
1525 3885 : is done so often.
1525 3886 :
1525 3887 : INPUTS: R10 Scratch
1525 3888 : R8 Node address
1525 3889 : R0 Scratch
1525 3890 :
1525 3891 : OUTPUTS: R10 NDI address if found, else 0
1525 3892 : R0 LBS if found
1525 3893 : LBC otherwise
1525 3894 :
1525 3895 : All other registers are unchanged
1525 3896 :
1525 3897 NET$NDI_BY_ADD::
1525 3898 : Get NDI by node address
1525 3899 : Save registers
1525 3900 : Get the RCB pointer
1525 3901 : Is this for area zero?
1525 3902 :
1525 3903 : Br if not, okay as it is
1525 3904 : Else, stuff our area into node address
1525 3905 :
1525 3906 10$: MOVL NET$GL_CNR_NDI,R11
1525 3907 : Point at root of NDI database
1525 3908 : Start at beginning of list
1525 3909 : Search for the right NDI
1525 3910 : Restore registers
1525 3911 : Br if success
1525 3912 : Get the local NDI CNF
1525 3913 : Get the RCB pointer
1525 3914 : Is this the local node?
1525 3915 : If EQL then yes
1525 3916 : Indicate failure
1525 3917 : Invalidate the NDI pointer
1525 3918 : Take common exit
1525 3919 15$: MOVL #1,R0
1525 3920 : Assume success
1525 3921 20$: RSB
```

50	00000000	8F	BB	1525	3898	PUSHR	#M<R8,R11>		
	00	58	06	1529	3899	MOVL	NET\$GL_PTR_VCB,R0		
		0A	ED	1530	3900	CMPZV	#TR4\$V_ADDR_AREA,-		
		07	12	1532	3901		#TR4\$S_ADDR_AREA,R8,#0		
	008B	C0	F0	1535	3902	BNEQ	10\$		
		0A		1537	3903	INSV	RCB\$B_HOMEAREA(R0),-		
	58	06		1538	3904		#TR4\$V_ADDR_AREA,-		
5B	00000000	8F	D0	153C	3905		#TR4\$S_ADDR_AREA,R8		
		5A	D4	153E	3906	10\$: MOVL	NET\$GL_CNR_NDI,R11		
				1545	3907	CLRL	R10		
	0900	8F	BA	1547	3908	\$SEARCH	eql,ndi,l,add		
	1A	50	E8	1556	3909	POPR	#M<R8,R11>		
5A	00000000	8F	D0	155A	3910	BLBS	R0,15\$		
50	00000000	8F	D0	155D	3911	MOVL	NET\$GL_LOCAL_NDI,R10		
	0E	A0	B1	1564	3912	MOVL	NET\$GL_PTR_VCB,R0		
		58	D4	1568	3913	CMPW	R8,RCB\$W_ADDR(R0)		
		06	13	156F	3914	BEQL	15\$		
		50	D4	1571	3915	CLRL	R0		
		5A	D4	1573	3916	CLRL	R10		
		03	11	1575	3917	BRB	20\$		
	50	01	D0	1577	3918	15\$: MOVL	#1,R0		
			05	157A	3919	20\$: RSB			

				157B	3921	.SBTTL NET\$LOCATE_NDI - Find phantom or real NDI CNF	
				157B	3922	:	
				157B	3923	:+ NET\$LOCATE_NDI - Find NDI CNF (phantom or real) by node address	
				157B	3924	:	
				157B	3925	: FUNCTIONAL DESCRIPTION:	
				157B	3926	:	
				157B	3927	: If an NDI entry exists for the specified node, it is returned. Otherwise,	
				157B	3928	: the address of a dummy NDI is returned as a 'phantom' NDI, so that the NDI	
				157B	3929	: block can be used on operations (such as event logging) for nodes that are	
				157B	3930	: reachable without being defined.	
				157B	3931	:	
				157B	3932	: INPUTS: R10 Scratch	
				157B	3933	: R8 Node address	
				157B	3934	: R0 Scratch	
				157B	3935	:	
				157B	3936	: OUTPUTS: R10 NDI address if found, else 0	
				157B	3937	: R0 LBS if found	
				157B	3938	: LBC otherwise	
				157B	3939	:	
				157B	3940	: All other registers are unchanged	
				157B	3941	: -	
				157B	3942	NET\$LOCATE_NDI::	; Get NDI by node address
				157B	3943	PUSHR #^M<R8,R11>	; Save registers
				157B	3944	MOVL NET\$G_L_PTR_VCB,R0	; Get the RCB pointer
				1586	3945	CMPZV #TR4\$V_ADDR_AREA,-	; Is this for area zero?
				1588	3946	#TR4\$S_ADDR_AREA,R8,#0	
				158B	3947	BNEQ 10\$	; Br if not, okay as it is
				158D	3948	INSV RCB\$B_HOMEAREA(R0),-	; Else, stuff our area into node address
				1591	3949	#TR4\$V_ADDR_AREA,-	
				1592	3950	#TR4\$S_ADDR_AREA,R8	
				1594	3951	10\$: MOVL NET\$G_L_CNR_NDI,R11	; Point at root of NDI database
				159B	3952	CLRL R10	; Start at beginning of list
				159D	3953	\$SEARCH eql,ndi,l,add	; Search for the right NDI
				15AC	3954	POPR #^M<R8,R11>	; Restore registers
				15B0	3955	BLBS R0,15\$	; Br if success
				15B3	3956	MOVL NET\$G_L_DUM_NDI,R10	; Return address of dummy NDI
				15BA	3957	MOVW R8,NDI_ADD(R10)	; Stuff the address
				15BE	3958	MOVW R8,CNF\$W_ID(R10)	; Here too
				15C2	3959	15\$: MOVL #1,R0	; Assume success
				15C5	3960	RSB	

	NET Sym	XWB XWB
	PSE ---	\$AB NET NET \$RM NET NET
	Pha --- Int Com Pas Syn Pas Syn Pse Cro Ass	The 24 The

```

15C6 3962 .SBTTL MOVE PARAMETER SUBROUTINES
15C6 3963 :
15C6 3964 : Subroutine to move CNF string field
15C6 3965 :
15C6 3966 MOVCTR:
EA37' 30 15C6 3967 BSBW CNF$GET_FIELD ; Set field descriptor
50 DD 15C9 3968 PUSHL R0 ; Save status
83 57 90 15CB 3969 MOV B R7,(R3)+ ; Enter count (could be zero)
05 11 15CE 3970 BRB MOVIT ; Move the string
15D0 3971 MOVSTR:
EA2D' 30 15D0 3972 BSBW CNF$GET_FIELD ; Set field descriptor
50 DD 15D3 3973 PUSHL R0 ; Save status
63 68 57 28 15D5 3974 MOVIT: MOV C3 R7,(R8),(R3) ; Move the string
50 8ED0 15D9 3975 POPL R0 ; Restore status
05 15DC 3976 RSB

```

```
15DD 3978 .SBTTL FMT_CNT - FORMAT COUNTERS
15DD 3979 :+
15DD 3980 FMT_CNT - Format counters
15DD 3981 :
15DD 3982 INPUTS: R6 Address of source counter block (FMT_CNT only)
15DD 3983 R5 Address of table to drive counter formatting
15DD 3984 R3 Address of next byte in output buffer
15DD 3985 R2 Number of bytes in source counter block
15DD 3986 R1 Pointer to source counter block (MOVE_FMT_CNT only)
15DD 3987 R0 Scratch
15DD 3988 :
15DD 3989 OUTPUTS: R3 Updated to next free byte in output buffer
15DD 3990 R2,R1 Garbage
15DD 3991 R0 $$$_NORMAL
15DD 3992 :
15DD 3993 All other registers are preserved
15DD 3994 :-
15DD 3995 .SAVE PSECT
00000056 3996 .PSECT NET_LOCK_CODE,NOWRT,GBL
0056 3997
0056 3998 MOVE_FMT CNT:: ; Move and format counters
5E 00000064 8F DD 0056 3999 PUSH R6 ; Save R6
56 5E D0 0058 4000 SUBL #CNT_FMT_BUFSIZ,SP ; Create work area on stack
0062 4001 MOVL SP,R6 ; Point to it with R6
0062 4002 :
0062 4003 ; Lock out NETDRIVER and take a snapshot of the counters. While
0062 4004 ; NETDRIVER is locked out, clear the counters if its called for.
0062 4005 :
0062 4006 DSBINT #NETSC_IPL ; Lock out Netdriver
66 61 52 BB 0068 4007 PUSHR #M<R1,R2,R3,R4,R5> ; Save regs
51 6E 7D 006A 4008 MOVCS R2,(R1),(R6) ; Move counters
10 00000000'EF 02 E1 0071 4009 MOVQ (SP),R1 ; Get R1,R2 (counter descriptor)
81 00000000'GF D0 0073 4010 BBC #NETSV CLR CNT, - ; If BC, don't clear the counters
52 00 6E 00 C2 0080 4011 NET$GL_FLAGS,20$ ;
61 52 00 6E 00 2C 0083 4012 MOVL G^EXESGL_ABSTIM,(R1)+ ; Reset Abs-time since last zeroed
3E BA 0089 4013 SUBL #4,R2 ; Adjust bytes left in counter block
08 10 008E 4014 MOVCS #0,(SP),#0,R2,(R1) ; Zero remaining counters
0090 4015 POPR #M<R1,R2,R3,R4,R5> ; Restore regs
0090 4016 ENBINT ; Restore IPL
0090 4017 BSBB FMT_CNT ; Format the counters
0090 4018 ; Done, restore the stack and return
5E 00000064 8F C0 0090 4019 ADDL #CNT_FMT_BUFSIZ,SP ; Create work area on stack
56 8ED0 0097 4020 POPL R6 ; Restore R6
05 009A 4021 RSB ; Return to caller
009B 4022
009B 4023 FMT_CNT:: ; Format counters
009B 4024 :
009B 4025 ; Move 'seconds since last zeroed' in NICE format to output buffer
66 00000000'GF 66 C3 009B 4026 SUBL3 (R6),G^EXESGL_ABSTIM,(R6) ; Get seconds since last zeroed
0000FFFF 8F 66 D1 00A3 4027 CMPL (R6),#X<FFFF> ; Has counter overflowed?
03 1B 00AA 4028 BLEQU 30$ ; If LEQU no
66 01 AE 00AC 4029 MNEGW #1,(R6) ; Latch counter at max value
83 C000 8F B0 00AF 4030 MOVW #NETSC_NMACNT_SLZ,(R3)+ ; Enter i.d. of counter
83 66 F7 00B4 4031 CVTLW (R6),(R3)+ ; Enter 'seconds since last zeroed'
00B7 4034 :
```

```

                                ; Move each counter one at a time in NICE format to the output buffer
                                ;
                                ;
52  85  B0 00B7 4035 40$: MOVW (R5)+,R2 ; Get the next NICE counter i.d.
27 13 00B7 4036 BEQL 100$ ; If EQL then done
51 85 3C 00BA 4038 MOVZWL (R5)+,R1 ; Get offset to counter value
51 56 C0 00BC 4039 ADDL R6,R1 ; Get pointer to counter value
83 52 B0 00BF 4040 MOVW R2,(R3)+ ; Enter NICE counter i.d.
52 52 02 0D EF 00C2 4041 EXTZV #13,#2,R2,R2 ; Get width of counter
                                ; DISPATCH R2,-
                                ; Dispatch on width
                                ;
                                ; <1, 70$>,- ; Byte
                                ; <2, 60$>,- ; Word
                                ; <3, 50$>,- ; Longword
                                ;
                                ;
                                ; BUG_CHECK NETNOSTATE,FATAL
                                ;
83 81 B0 00D8 4050 50$: MOVW (R1)+,(R3)+ ; Counter is a longword
83 81 90 00DB 4051 60$: MOVW (R1)+,(R3)+ ; Counter is a word
83 81 90 00DE 4052 70$: MOVW (R1)+,(R3)+ ; Counter is a byte
D4 11 00E1 4053 BRB 40$ ; Loop
                                ;
                                ; Done
                                ;
50 0000'8F 3C 00E3 4054 100$: ;
05 00E3 4055 ;
000015DD 00E3 4056 ;
00E3 4057 ;
00E3 4058 ;
00E8 4059 ; Always successful
00E9 4060 ; Return
000015DD 4061 .RESTORE_PSECT
```

```
15DD 4063 .SBTTL LOG_COUNTERS - LOG ZERO COUNTER EVENT
15DD 4064 :+
15DD 4065 : LOG_COUNTERS - Conditionally log zero counter event
15DD 4066 :
15DD 4067 : INPUTS: R11 CNR pointer
15DD 4068 : R10 CNF pointer
15DD 4069 : R5 Scratch
15DD 4070 : R3 Address of first byte past counter block
15DD 4071 : R2 Address of counter block
15DD 4072 : R0 EVC database i.d.
15DD 4073 :
15DD 4074 : OUTPUT: R5,R0 Garbage
15DD 4075 :
15DD 4076 : All other registers are preserved.
15DD 4077 :-
15DD 4078 LOG_COUNTERS:: : Conditionally log zero counter event
15DD 4079 BBC #NETSV_CLRCNT, - : If BC then counters weren't zeroed
15DF 4080 NET$GL_FLAGS, 20$ :
15E5 4081 MOVAB NET$AB_EVT_WQE, R5 : Point to the common WQE
15EC 4082 MOVW CNF$W_ID(RT0), - :
15EF 4083 WQESW-REQIDT(R5) : Setup the CNF i.d.
15F1 4084 MOVL R2, WQESL_EVL_PKT(R5) : Setup pointer to counter block
15F5 4085 SULB3 R2, R3, WQESB_EVL_DT2(R5) : Setup size of counter block
15FA 4086 MOVW R0, WQESB_EVC_DTT(R5) : Setup database i.d.
15FE 4087 MOVW #EVCSC_NSL_DBR, - : Assume data base re-used event
1602 4088 WQESW_EVL_CODE(R5) :
1604 4089 BBS #NETSV_LOGDBR, - : If BS then data base re-used event
1606 4090 NET$GL_FLAGS, 10$ :
160C 4091 MOVW #EVCSC_NMA_ZER, - : Else, assume zero counters event
160E 4092 WQESW_EVL_CODE(R5) :
1610 4093 BBC #NETSV_TIMER, - : If BC then zero counters event
1612 4094 NET$GL_FLAGS, 10$ :
1618 4095 MOVW #EVCSC_NMA_CTR, - : Else counter timer event
161A 4096 WQESW_EVL_CODE(R5) :
161C 4097 10$: BSBW NET$EVT_INTRAW : Log it
161F 4098 20$: RSB : Done
1620 4099
1620 4100
1620 4101 .END
```

02 E1
3A 00000000'EF
55 00000000'EF 9E
12 AA B0
12 A5
18 A5 52 D0
1F A5 53 52 83
1E A5 50 90
00C2 8F B0
1C A5
01 E0
10 00000000'EF 1604
09 B0 1606
1C A5 160C
04 E1 160E
04 1610
04 1612
08 B0 1618
1C A5 161A
E9E1' 30 161C
05 161F
1620 4099
1620 4100
1620 4101 .END

NETCNFACT
Symbol table

H 14
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 92
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

\$\$TAB	= 000000E0	R	02	CNFSV_FLG_MRK3	= 00000006		
\$\$TABEND	= 00000130	R	02	CNFSW_ID	= 00000012		
\$\$TMP	= 00000000			CNFS_ADVANCE	= 00000000		
\$\$TMP1	= 00000001			CNFS_QUIT	= 00000002		
\$\$TMP2	= 00000050			CNFS_TAKE_CURR	= 00000003		
\$\$TMPX	= 00000000	R	04	CNFS_TAKE_PREV	= 00000001		
\$\$TMPX1	= 0000000F			CNFADD	= 0000001C		
\$\$T1	= 00000001			CNR\$B_FLG	= 0000000B		
\$\$NSPMSG	= 00000000			CNR\$B_TYPE	= 0000000A		
\$\$TR3MSG	= 00000000			CNR\$B_BLINK	= 00000004		
\$\$TR4MSG	= 00000000			CNR\$B_FLD_COLL	= 00000014		
\$WIDTH_B	= 00000001			CNR\$B_FLINK	= 00000000		
\$WIDTH_L	= 00000003			CNR\$W_SIZ_CNF	= 0000000C		
\$WIDTH_W	= 00000002			CNT_FMT_BUF\$IZ	= 00000064		
ACP\$C_STA_F	= 00000004			CONVERT	00001138	R	05
ACP\$C_STA_H	= 00000005			DEFAULT_SCAN	00000000	RG	05
ACP\$C_STA_I	= 00000000			DW\$B_S\$UBSTA	= 00000046		
ACP\$C_STA_N	= 00000001			DW\$B\$-REM\$OD	= 00000040		
ACP\$C_STA_R	= 00000002			DW\$B\$L-PID	= 00000034		
ACP\$C_STA_S	= 00000003			DW\$B\$W-ID	= 0000004E		
ADJ\$B_BCPRI	= 0000000C			DW\$B\$W-PATH	= 0000003E		
ADJ\$B_PTYPE	= 00000001			EVC\$C_NMA_CTR	= 00000008		
ADJ\$B_STS	= 00000000			EVC\$C_NMA_ZER	= 00000009		
ADJ\$C_PTY_AREA	= 00000003			EVC\$C_NSL_DBR	= 000000C2		
ADJ\$C_PTY_PH2	= 00000002			EVC\$C_SRC_NOD	= 00000000		
ADJ\$C_PTY_PH4N	= 00000005			EX\$G\$L_AB\$TIM	*****	X	06
ADJ\$C_PTY_UNK	= FFFFFFFF			EX\$E\$IPID_TO_EPID	*****	X	05
ADJ\$V_INUSE	= 00000000			FAB\$B_DNS	= 00000035		
ADJ\$V_RUN	= 00000001			FAB\$B_FNS	= 00000034		
ADJ\$W_BUF\$IZ	= 00000006			FAB\$C_BID	= 00000003		
ADJ\$W_INT_LSN	= 00000008			FAB\$C_BLN	= 00000050		
ADJ\$W_PNA	= 00000004			FAB\$C_SEQ	= 00000000		
AREA_COST_HOPS	00001438	R	05	FAB\$C_VAR	= 00000002		
BIT...	= 00000006			FAB\$C_ALQ	= 00000010		
BTECOR	= 00000008			FAB\$C_DNA	= 00000030		
BUG\$ NETNOSTATE	*****	X	05	FAB\$C_FNA	= 0000002C		
CALLER	= 00000004			FAB\$C_FOP	= 00000004		
CALL_NETDRIVER	*****	X	05	FAB\$V_CHAN_MODE	= 00000002		
CHK_LOGIN_NDI	00000A1B	R	05	FAB\$V_FILE_MODE	= 00000004		
CHK_LOGIN_OBI	00000A10	R	05	FAB\$V_LNM_MODE	= 00000000		
CLU\$GL_CLUB	*****	X	05	FAB\$W_GBC	= 00000048		
CLUNODE_DESC	00000010	R	03	FMT_CNT	0000009B	RG	06
CNFSB_FLG	= 0000000B			GET_COLLATE	000002A0	R	05
CNFS\$C-LENGTH	= 00000024			GET_COST_HOPS	00000CF1	R	05
CNFS\$GET_FIELD	*****	X	05	GET_DWB	00000381	R	05
CNFS\$INIT	*****	X	05	GET_DWB_ADR	00001398	R	05
CNFS\$KEY_SEARCH	*****	X	05	GET_JPI	0000122F	R	05
CNFS\$L_BLINK	= 00000004			HACT	00000EE6	R	05
CNFS\$L_FLINK	= 00000000			ICB\$C_ACCESS	= 00000040		
CNFS\$L_MASK	= 00000018			IOSB	00000038	R	02
CNFS\$M_FLG_DELETE	= 00000002			ITEM_LIST	00000064	R	02
CNFS\$POT_F\$ELD	*****	X	05	JPI\$-PRCNAM	= 0000031C		
CNFSV_FLG_ACP	= 00000002			JPI\$-USERNAME	= 00000202		
CNFSV_FLG_CNR	= 00000000			LLIS\$C_XWB	= 00000000		
CNFSV_FLG_DELETE	= 00000001			LNIS\$B_STA	= 00000002		
CNFSV_FLG_MRK1	= 00000004			LNIS\$C_STA_INIT	= 00000004		
CNFSV_FLG_MRK2	= 00000005			LNIS\$C_STA_OFF	= 00000001		

NE
VO

NETCNFACT
Symbol table

I 14
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 93
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

LNISW_ADD = 00000000
LOCAL_NODE_CNT = 00000E16 R 05
LOCATE_ADJ = 0000138C R 05
LOG_COUNTERS = 000015DD RG 05
LPDSC_LOC_INX = 00000001
LSB = 00000000
LSBSB_R_CXBCNT = 00000028
LSBSB_R_CXBQUO = 00000029
LSBSB_SPARE = 0000002A
LSBSB_STS = 0000002B
LSBSB_X_ADJ = 0000000B
LSBSB_X_CXBACT = 0000000D
LSBSB_X_CXBCNT = 0000000F
LSBSB_X_CXBQUO = 0000000E
LSBSB_X_PKTWND = 0000000C
LSBSB_X_REQ = 0000000A
LSBSL_CROSS = 0000002C
LSBSL_R_CXB = 00000020
LSBSL_R_IRP = 0000001C
LSBSL_X_CXB = 00000018
LSBSL_X_IRP = 00000014
LSBSL_X_PND = 00000010
LSBSM_BOM = 00000020
LSBSM_EOM = 00000040
LSBSM_LI = 00000001
LSBSL_LSB = 00000030
LSBSL_SPARE = 00000004
LSBSL_STS = 00000001
LSBSV_BOM = 00000005
LSBSV_EOM = 00000006
LSBSV_LI = 00000000
LSBSV_SPARE = 00000001
LSBSW_HAA = 00000008
LSBSW_HAR = 00000006
LSBSW_HAX = 00000026
LSBSW_HNR = 00000024
LSBSW_HXS = 00000004
LSBSW_LNX = 00000002
LSBSW_LUX = 00000000
LTBSL_SLOTS = 00000010
LTBSW_SLT_TOT = 00000004
MOVCTR = 000015C6 R 05
MOVE_FMT_CNT = 00000056 RG 06
MOVIT = 000015D5 R 05
MOVSTR = 000015D0 R 05
NAMS_B_ESL = 0000000B
NAMS_B_ESS = 0000000A
NAMS_B_NOP = 00000008
NAMS_B_RSS = 00000002
NAMS_C_BID = 00000002
NAMS_C_BLN = 00000060
NAMS_L_ESA = 0000000C
NAMS_L_RSA = 00000004
NDCSC_LENGTH = 0000001C
NDCSL_ABS_TIM = 00000000
NDCSL_BRC = 0000000C
NDCSL_BSN = 00000010

NDCSL_MRC = 00000014
NDCSL_MSX = 00000018
NDCSL_PRC = 00000014
NDCSL_PSN = 00000018
NDCSW_CRC = 00000008
NDCSW_CSN = 0000000A
NDCSW_RSE = 00000004
NDCSW_RTO = 00000006
NDC_CRT_TAB = 00000058 R 03
NDISW_ADD = 00000000
NDIDEF_SCAN = 00000031 RG 05
NDI_ADD = 00000024
NDI_LNAMEBUF = 00000028 R 02
NDI_L_NACS = 00000000 R 02
NDI_L_TAD = 00000078 R 05
NDI_MARKER = 0000071D R 05
NDI_NLOGIN_VEC = 00000028 R 03
NDI_PLOGIN_VEC = 00000038 R 03
NDI_Q_LNAME = 00000020 R 02
NDI_Q_NAME = 00000018 R 02
NDI_SETUP = 00000EF4 R 05
NDI_V_LOCAL = 00000005
NDI_V_LOOP = 00000004
NDI_V_MARKER = 00000006
NDI_Z_COL = 00000004 R 02
NETSAB_EVT_WQE = ***** X 05
NETSADD_NDI = ***** X 05
NETSADJ_LPD_CRI = ***** X 05
NETSAJI_L_ADD = 000012EF RG 05
NETSAJI_L_BLO = 00001320 RG 05
NETSAJI_L_LIT = 0000130D RG 05
NETSAJI_L_RPR = 00001333 RG 05
NETSAJI_L_TYP = 00001302 RG 05
NETSAJI_S_CIR = 0000137C RG 05
NETSAJI_S_COL = 00001347 RG 05
NETSAJI_S_NNA = 00001355 RG 05
NETSAJI_V_LCK = 000012DF RG 05
NETSAJI_V_REA = 000012E3 RG 05
NETSALLOCATE = ***** X 05
NETSAL_CNF_DFLT = ***** X 05
NETSAPPLY_DFLT = 00000567 RG 05
NETSAREA_REACH = 000014B1 RG 05
NETSARI_C_ADD = 00001414 RG 05
NETSARI_L_DCO = 0000141C RG 05
NETSARI_L_DHO = 0000142A RG 05
NETSARI_L_NND = 0000146D RG 05
NETSARI_S_COL = 00001488 RG 05
NETSARI_S_DLI = 00001496 RG 05
NETSARI_V_LCK = 00001402 RG 05
NETSARI_V_REA = 00001406 RG 05
NETSAW_AREA_C_H = ***** X 05
NETSAW_MIN_C_H = ***** X 05
NETSC_ACT_TIMER = 0000001E
NETSC_DR_THIRD = 00000008
NETSC_EFN_ASYN = 00000002
NETSC_EFN_WAIT = 00000001
NETSC_IPL = 00000008

NETCNFACT
Symbol table

J 14
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 94
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

NETSC_MAXACCFD	= 00000027		NET\$GL_DUM_NDI	*****	X	05	
NETSC_MAXLINNAM	= 0000000F		NET\$GL_FLAGS	*****	X	05	
NETSC_MAXLNK	= 000003FF		NET\$GL_LOCAL_NDI	*****	X	05	
NETSC_MAXNODNAM	= 00000006		NET\$GL_NET_UCB	*****	X	05	
NETSC_MAXOBJNAM	= 0000000C		NET\$GL_PTR_LNI	*****	X	05	
NETSC_MAX_AREAS	= 0000003F		NET\$GL_PTR_VCB	*****	X	05	
NETSC_MAX_LINES	= 00000040		NET\$GL_SRCH_ID	*****	X	05	
NETSC_MAX_NCB	= 0000006E		NET\$GQ_SRCH_KEY	*****	X	05	
NETSC_MAX_NODES	= 000003FF		NET\$GQ_TMP_BUF	*****	X	05	
NETSC_MAX_OBJ	= 000000FF		NET\$G CNI AREA	*****	X	05	
NETSC_MAX_WQE	= 00000014		NET\$INSERT_AJI	000009F5	RG	05	
NETSC_MINBUFSIZ	= 000000C0		NET\$INSERT_ARI	00000A07	RG	05	
NETSC_NMACNT_SLZ	= 0000C000		NET\$INSERT_EFI	000009EE	RG	05	
NETSC_TID_ACT	= 00000003		NET\$INSERT_ESI	000009EB	RG	05	
NETSC_TID_RUS	= 00000001		NET\$INSERT_LLI	000009F1	RG	05	
NETSC_TID_XRT	= 00000002		NET\$INSERT_LNI	000005B8	RG	05	
NETSC_TRCTL_CEL	= 00000002		NET\$INSERT_NDI	000007AD	RG	05	
NETSC_TRCTL_OVR	= 00000005		NET\$INSERT_OBI	000009C1	RG	05	
NETSC_UTLBUFSIZ	= 00001000		NET\$INSERT_SDI	000009FE	RG	05	
NET\$DBC_EFI	*****	X	05	NET\$INSERT_SPI	000009F1	RG	05
NET\$DBC_ESI	*****	X	05	NET\$LLI_L_DLY	0000118F	PG	05
NET\$DEACLOCATE	*****	X	05	NET\$LLI_L_IPID	00001182	RG	05
NET\$DEFAULT_AJI	00000567	RG	05	NET\$LLI_L_LLN	000011A9	RG	05
NET\$DEFAULT_ARI	00000567	RG	05	NET\$LLI_L_PID	0000116C	RG	05
NET\$DEFAULT_EFI	00000567	RG	05	NET\$LLI_L_PNA	000011B6	RG	05
NET\$DEFAULT_ESI	00000567	RG	05	NET\$LLI_L_RLN	0000119C	RG	05
NET\$DEFAULT_LLI	00000567	RG	05	NET\$LLI_L_STA	000011C3	RG	05
NET\$DEFAULT_LNI	00000548	RG	05	NET\$LLI_S_CNT	000011D0	RG	05
NET\$DEFAULT_NDI	0000058A	RG	05	NET\$LLI_S_COL	00001270	RG	05
NET\$DEFAULT_OBI	00000567	RG	05	NET\$LLI_S_PNN	00001293	RG	05
NET\$DEFAULT_SDI	00000567	RG	05	NET\$LLI_S_PRC	000011E9	RG	05
NET\$DEFAULT_SPI	00000567	RG	05	NET\$LLI_S_RID	000011D5	RG	05
NET\$DELETE_AJI	00000AB0	RG	05	NET\$LLI_S_USR	0000120C	RG	05
NET\$DELETE_ARI	00000AB0	RG	05	NET\$LLI_V_LCK	0000114D	RG	05
NET\$DELETE_BTE	*****	X	05	NET\$LNI_L_ACL	00000C09	RG	05
NET\$DELETE_EFI	00000B07	RG	05	NET\$LNI_L_ADD	00000BF5	RG	05
NET\$DELETE_ESI	00000AF2	RG	05	NET\$LNI_L_STA	00000BDC	RG	05
NET\$DELETE_LLI	00000B1C	RG	05	NET\$LNI_S_CNT	00000C4B	RG	05
NET\$DELETE_LNI	00000AB0	RG	05	NET\$LNI_S_COL	00000C1C	RG	05
NET\$DELETE_NDI	00000AB3	RG	05	NET\$LNI_S_NAM	00000C24	RG	05
NET\$DELETE_OBI	00000AE2	RG	05	NET\$LNI_S_PHA	00000C5D	RG	05
NET\$DELETE_SDI	00000AB0	RG	05	NET\$LNI_V_LCK	00000BC4	RG	05
NET\$DELETE_SPI	00000B3E	RG	05	NET\$LOCATE_NDI	0000157B	RG	05
NET\$EFI_S_COL	00001131	RG	05	NET\$M_MAXLNKMSK	= 000003FF		
NET\$EFI_V_LCK	0000112B	RG	05	NET\$NDI_BY_ADD	00001525	RG	05
NET\$ESI_S_COL	00001122	RG	05	NET\$NDI_L_ACL	00000D41	RG	05
NET\$ESI_V_LCK	00001104	RG	05	NET\$NDI_L_ADD	00000CC7	RG	05
NET\$EVT_INTRA	*****	X	05	NET\$NDI_L_DCO	00000CDB	RG	05
NET\$FIND_ADJ	*****	X	05	NET\$NDI_L_DEL	00000D4E	RG	05
NET\$FIND_NAME	*****	X	05	NET\$NDI_L_DHO	00000CE6	RG	05
NET\$FIND_NDI	*****	X	05	NET\$NDI_L_DTY	00000D5B	RG	05
NET\$GET_END	*****	X	05	NET\$NDI_L_NND	00000D88	RG	05
NET\$GET_LOC_STA	000014F7	RG	05	NET\$NDI_L_TAD	00000D75	RG	05
NET\$GET_LPD_CRI	*****	X	05	NET\$NDI_S_CNT	00000DE7	RG	05
NET\$GL_CNR_CNI	*****	X	05	NET\$NDI_S_COL	00000EC4	RG	05
NET\$GL_CNR_NDI	*****	X	05	NET\$NDI_S_DLI	00000E89	RG	05
NET\$GL_DLE_UCBO	*****	X	05	NET\$NDI_S_HAC	00000EE2	RG	05

NETCNFACT
Symbol table

K 14
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 95
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

NET\$NDI_S_NNM	00000F11	RG	05	NET\$SHOW_OBI	00000544	RG	05
NET\$NDI_V_LCK	00000C86	RG	05	NET\$SHOW_SDI	00000544	RG	05
NET\$NDI_V_LOO	00000C78	RG	05	NET\$SHOW_SPI	00000544	RG	05
NET\$NDI_V_REA	00000C8C	RG	05	NET\$SPCINS_AJI	00000A50	RG	05
NET\$OBI_S_COL	00000FF3	RG	05	NET\$SPCINS_ARI	00000A50	RG	05
NET\$OBI_S_IAC	0000101C	RG	05	NET\$SPCINS_CRI	00000A50	RG	05
NET\$OBI_S_SFI	0000107B	RG	05	NET\$SPCINS_DEF	00000A50	RG	05
NET\$OBI_S_ZNA	00000FF3	RG	05	NET\$SPCINS_EFI	00000A50	RG	05
NET\$OBI_V_LCK	00000FDF	RG	05	NET\$SPCINS_ESI	00000A50	RG	05
NET\$PRE_QIO_AJI	00000540	RG	05	NET\$SPCINS_LLI	00000A50	RG	05
NET\$PRE_QIO_ARI	00000540	RG	05	NET\$SPCINS_LNI	00000A50	RG	05
NET\$PRE_QIO_EFI	00000540	RG	05	NET\$SPCINS_NDI	00000A86	RG	05
NET\$PRE_QIO_ESI	00000540	RG	05	NET\$SPCINS_OBI	00000A50	RG	05
NET\$PRE_QIO_LLI	00000540	RG	05	NET\$SPCINS_PLI	00000A50	RG	05
NET\$PRE_QIO_LNI	00000540	RG	05	NET\$SPCINS_SDI	00000A50	RG	05
NET\$PRE_QIO_NDI	000004FE	RG	05	NET\$SPCINS_SPI	00000A50	RG	05
NET\$PRE_QIO_OBI	00000540	RG	05	NET\$SPCSCAN_AJI	0000042E	RG	05
NET\$PRE_QIO_SDI	00000540	RG	05	NET\$SPCSCAN_ARI	0000042E	RG	05
NET\$PRE_QIO_SPI	00000540	RG	05	NET\$SPCSCAN_CRI	0000042E	RG	05
NET\$READ_NDI_CNT	*****	X	05	NET\$SPCSCAN_EFI	0000042E	RG	05
NET\$REMOVE_AJI	00000B42	RG	05	NET\$SPCSCAN_ESI	0000042E	RG	05
NET\$REMOVE_ARI	00000B42	RG	05	NET\$SPCSCAN_LLI	0000042E	RG	05
NET\$REMOVE_DEF	00000B42	RG	05	NET\$SPCSCAN_LNI	0000042E	RG	05
NET\$REMOVE_EFI	00000B94	RG	05	NET\$SPCSCAN_NDI	00000431	RG	05
NET\$REMOVE_ESI	00000B94	RG	05	NET\$SPCSCAN_OBI	0000042E	RG	05
NET\$REMOVE_LLI	00000B69	RG	05	NET\$SPCSCAN_PLI	0000042E	RG	05
NET\$REMOVE_LNI	00000B42	RG	05	NET\$SPCSCAN_SDI	0000042E	RG	05
NET\$REMOVE_NDI	00000BC3	RG	05	NET\$SPCSCAN_SPI	0000042E	RG	05
NET\$REMOVE_OBI	00000B42	RG	05	NET\$SPI_S_COL	000012C5	RG	05
NET\$REMOVE_SDI	00000B42	RG	05	NET\$SPI_V_LCK	000012C1	RG	05
NET\$REMOVE_SPI	00000B42	RG	05	NET\$TBLCE_DFLT	00000573	RG	05
NET\$RESUME_NDI	*****	X	05	NET\$TEST_REACH	00000F6F	RG	05
NET\$SCAN_AJI	000002D4	RG	05	NET\$TRAVERSE_ALT	*****	X	05
NET\$SCAN_ARI	000003C7	RG	05	NET\$TRAVERSE_NDI	*****	X	05
NET\$SCAN_EFI	00000000	RG	05	NET\$T_CNF_AJI	*****	X	05
NET\$SCAN_ESI	00000000	RG	05	NET\$T_CNF_ARI	*****	X	05
NET\$SCAN_LLI	00000000	RG	05	NET\$T_CNF_SDI	*****	X	05
NET\$SCAN_LNI	00000000	RG	05	NET\$T_PRSNAM	00000080	R	02
NET\$SCAN_NDI	000000FC	RG	05	NET\$T_SYSFAB	000000E0	R	02
NET\$SCAN_OBI	00000000	RG	05	NET\$UPD_LOCAL	*****	X	05
NET\$SCAN_SDI	00000332	RG	05	NET\$V_CLCNT	= 00000002		
NET\$SCAN_SPI	00000000	RG	05	NET\$V_INTRNL	= 00000009		
NET\$SDI_C_PID	000013A7	RG	05	NET\$V_LOGDBR	= 00000001		
NET\$SDI_L_SUB	0000139D	RG	05	NET\$V_TIMER	= 00000004		
NET\$SDI_S_CIR	000013BF	RG	05	NETUPD\$_DSCLNK	= 00000009		
NET\$SDI_S_COL	000013B1	RG	05	NEXT_HOP_ADJ	00000D98	R	05
NET\$SDI_S_PHA	000013D6	RG	05	NFBSC_CRT_NAM	= 04020041		
NET\$SDI_S_PRC	000013E4	RG	05	NFBSC_DB_AJI	= 00000013		
NET\$SDI_V_LCK	00001394	RG	05	NFBSC_DB_ARI	= 00000014		
NET\$SET_CTR_TIMER	*****	X	05	NFBSC_DB_SDI	= 0000001A		
NET\$SHOW_AJI	00000544	RG	05	NFBSC_EFI_LCK	= 06000001		
NET\$SHOW_ARI	00000544	RG	05	NFBSC_EFI_SIN	= 06010010		
NET\$SHOW_EFI	00000544	RG	05	NFBSC_ESI_LCK	= 07000001		
NET\$SHOW_ESI	00000544	RG	05	NFBSC_ESI_SNK	= 07010010		
NET\$SHOW_LLI	00000544	RG	05	NFBSC_ESI_STA	= 07010011		
NET\$SHOW_LNI	00000544	RG	05	NFBSC_LLI_STA	= 08010011		
NET\$SHOW_NDI	00000544	RG	05	NFBSC_LNI_ADD	= 01010010		

NETCNFACT
Symbol table

- Configuration data base access action

L 14

16-SEP-1984 01:13:22 VAX/VMS Macro V04-00
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1

Page 96
(48)

NFBSC_LNI_ETY = 0101001A
NFBSC_LNI_MAD = 0101001E
NFBSC_LNI_MAR = 0101002D
NFBSC_LNI_STA = 01010014
NFBSC_LNI_SUP = 01000002
NFBSC_NDI_ADD = 02010012
NFBSC_NDI_COL = 02020040
NFBSC_NDI_NAC = 02020052
NFBSC_NDI_NLI = 0202004C
NFBSC_NDI_NNA = 02020043
NFBSC_NDI_NPW = 02020053
NFBSC_NDI_NUS = 02020051
NFBSC_NDI_PAC = 0202004F
NFBSC_NDI_PPW = 02020050
NFBSC_NDI_PUS = 0202004E
NFBSC_NDI_TAD = 02010010
NFBSC_OBI_ACC = 03020047
NFBSC_OBI_FID = 03020045
NFBSC_OBI_NAM = 03020044
NFBSC_OBI_NUM = 03010014
NFBSC_OBI_PID = 03010015
NFBSC_OBI_PSW = 03020048
NFBSC_OBI_SET = 03000002
NFBSC_OBI_UCB = 03010012
NFBSC_OBI_USR = 03020046
NFBSC_OP_EQL = 00000000
NFBSC_OP_FNDPOS = 00000006
NFBSC_SPT_PID = 12010010
NMASC_CTNOB_APL = 00000384
NMASC_CTNOB_BRC = 00000258
NMASC_CTNOB_BSN = 00000259
NMASC_CTNOB_CRC = 0000026C
NMASC_CTNOB_CSN = 0000026D
NMASC_CTNOB_MLL = 0000028C
NMASC_CTNOB_MRC = 00000262
NMASC_CTNOB_MSN = 00000263
NMASC_CTNOB_NOL = 00000386
NMASC_CTNOB_NUL = 00000385
NMASC_CTNOB_OPL = 00000387
NMASC_CTNOB_PFE = 0000038E
NMASC_CTNOB_RSE = 00000280
NMASC_CTNOB_RTO = 00000276
NMASC_CTNOB_RUL = 00000398
NMASC_CTNOB_VER = 000003A2
NMASC_STATE_OFF = 00000001
NODADD = 00000020
NODE_CNT = 00000E46 R 05
NOT_LOOPNODE = 0000087D R 05
NSP\$\$\$_QUAL_ACK = 00000000
NSP\$\$\$_QUAL_ALTFLW = 00000000
NSP\$\$\$_QUAL_DATA = 00000000
NSP\$\$\$_QUAL_FLW = 00000000
NSP\$\$\$_QUAL_INF = 00000000
NSP\$\$\$_QUAL_MSG = 00000000
NSP\$\$\$_QUAL_SRV = 00000000
NSPSC_EXT_LNK = 0000001E
NSPSC_FLW_DATA = 00000000

NSPSC_FLW_INT = 00000001
NSPSC_FLW_NOP = 00000000
NSPSC_FLW_XOFF = 00000001
NSPSC_FLW_XON = 00000002
NSPSC_HSZ_ACK = 00000007
NSPSC_HSZ_CA = 00000003
NSPSC_HSZ_CC = 00000064
NSPSC_HSZ_CD = 000000F0
NSPSC_HSZ_CI = 000000F0
NSPSC_HSZ_DATA = 00000009
NSPSC_HSZ_DC = 00000016
NSPSC_HSZ_DI = 00000016
NSPSC_HSZ_INT = 00000009
NSPSC_HSZ_LS = 00000009
NSPSC_INF_V31 = 00000001
NSPSC_INF_V32 = 00000000
NSPSC_INF_V33 = 00000002
NSPSC_MAXHDR = 00000009
NSPSC_MSG_CA = 00000024
NSPSC_MSG_CC = 00000028
NSPSC_MSG_CI = 00000018
NSPSC_MSG_DATA = 00000000
NSPSC_MSG_DC = 00000048
NSPSC_MSG_DI = 00000038
NSPSC_MSG_DTACK = 00000004
NSPSC_MSG_INT = 00000030
NSPSC_MSG_LIACK = 00000014
NSPSC_MSG_LS = 00000010
NSPSC_SRV_MFC = 00000002
NSPSC_SRV_NFC = 00000000
NSPSC_SRV_REQ = 00000001
NSPSC_SRV_SFC = 00000001
NSPSC_ACK_NAK = 00001000
NSPSC_ACK_NUM = 00000FFF
NSPSC_ACK_VALID = 00008000
NSPSC_DATA_BOM = 00000020
NSPSC_DATA_EOM = 00000040
NSPSC_DATA_OVFW = 00000080
NSPSC_FLW_CHAN = 0000000C
NSPSC_FLW_DRV = 000000F0
NSPSC_FLW_INT = 00000020
NSPSC_FLW_INUSE = 00000010
NSPSC_FLW_LISUB = 00000004
NSPSC_FLW_MODE = 00000003
NSPSC_FLW_SP1 = 00000008
NSPSC_FLW_SP2 = 00000040
NSPSC_FLW_SP3 = 00000080
NSPSC_FLW_XOFF = 00000001
NSPSC_FLW_XON = 00000002
NSPSC_INF_VER = 00000003
NSPSC_MSG_INT = 00000020
NSPSC_MSG_LI = 00000010
NSPSC_SRV_01 = 00000003
NSPSC_SRV_EXT = 00000080
NSPSC_SRV_FLW = 0000000C
NSPSC_SRV_REQ = 000000F3
NSPSC_SRV_SP1 = 00000070

NETCNFACT
Symbol table

M 14
- Configuration data base access action

16-SEP-1984 01:13:22 VAX/VMS Macro V04-00
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1

Page 97
(48)

NSP\$R_QUAL = 00000000
NSP\$S_ACK_NUM = 0000000C
NSP\$S_ACK_SP2 = 00000002
NSP\$S_DATA_SP = 00000005
NSP\$S_FLW_CHAN = 00000002
NSP\$S_FLW_DRV = 00000004
NSP\$S_FLW_MODE = 00000002
NSP\$S_INF_VER = 00000002
NSP\$S_MSG_SP1 = 00000004
NSP\$S_NSPMSG = 00000005
NSP\$S_QUAL = 00000005
NSP\$S_QUAL_ACK = 00000002
NSP\$S_QUAL_ALTFLW = 00000001
NSP\$S_QUAL_DATA = 00000001
NSP\$S_QUAL_FLW = 00000001
NSP\$S_QUAL_INF = 00000001
NSP\$S_QUAL_MSG = 00000005
NSP\$S_QUAL_SRV = 00000001
NSP\$S_SRV_01 = 00000002
NSP\$S_SRV_FLW = 00000002
NSP\$S_SRV_SP1 = 00000003
NSP\$V_ACK_NAK = 0000000C
NSP\$V_ACK_NUM = 00000000
NSP\$V_ACK_SP2 = 0000000D
NSP\$V_ACK_VALID = 0000000F
NSP\$V_DATA_BOM = 00000005
NSP\$V_DATA_EOM = 00000006
NSP\$V_DATA_OVFW = 00000007
NSP\$V_DATA_SP = 00000000
NSP\$V_FLW_CHAN = 00000002
NSP\$V_FLW_DRV = 00000004
NSP\$V_FLW_INT = 00000005
NSP\$V_FLW_INUSE = 00000004
NSP\$V_FLW_LISUB = 00000002
NSP\$V_FLW_MODE = 00000000
NSP\$V_FLW_SP1 = 00000003
NSP\$V_FLW_SP2 = 00000006
NSP\$V_FLW_SP3 = 00000007
NSP\$V_FLW_XOFF = 00000000
NSP\$V_FLW_XON = 00000001
NSP\$V_INF_VER = 00000000
NSP\$V_MSG_INT = 00000005
NSP\$V_MSG_LI = 00000004
NSP\$V_MSG_SP1 = 00000000
NSP\$V_SRV_03 = 00000000
NSP\$V_SRV_EXT = 00000007
NSP\$V_SRV_FLW = 00000002
NSP\$V_SRV_SP1 = 00000004
NSP\$W_DSTLNK = 00000001
NSP\$W_SRCLNK = 00000003
OBI_LOGIN_VEC = 00000048 R 03
ORIGAP = 00000000
PNAME = 00000054 R 02
PNAME\$ = 00000050 R 02
PR\$ IPL = ***** X 06
RCB\$B_CNT_APL = 00000095
RCB\$B_CNT_NOL = 00000094

RCB\$B_CNT_OPL = 00000096
RCB\$B_CNT_PFE = 00000097
RCB\$B_CNT_RUL = 00000098
RCB\$B_CNT_VER = 00000099
RCB\$B_ETY = 0000008A
RCB\$B_HOMEAREA = 0000008B
RCB\$B_MAX_AREA = 0000008C
RCB\$B_STATUS = 0000000B
RCB\$C_CNT_SIZE = 0000000C
RCB\$C_ABS_TIM = 00000090
RCB\$C_PTR_ADJ = 0000002C
RCB\$C_PTR_AOA = 00000020
RCB\$C_PTR_LTB = 00000024
RCB\$C_PTR_OA = 0000001C
RCB\$V_LVL2 = 00000000
RCB\$W_ADDR = 0000000E
RCB\$W_ALIAS = 0000008D
RCB\$W_CNT_MLL = 0000009E
RCB\$W_CNT_NUL = 0000009A
RCB\$W_DRT = 000000AA
RCB\$W_LVL2 = 000000AC
RCB\$W_MAX_ADDR = 000000FA
RCB\$W_MAX_ADJ = 00000063
RCB\$W_MCOINT = 00000054
RCB_CNT_TAB = 0000007C R 03
SAVREG = 0000000C
SCAN_XWB = 00000000 R 06
SCS\$GB_NODENAME = ***** X 05
SCS\$GB_SYSTEMID = ***** X 05
SIZ... = 00000001
SS\$BADPARAM = ***** X 05
SS\$DEACTIVE = ***** X 05
SS\$ILLCNTRFUNC = ***** X 05
SS\$INSFARG = ***** X 05
SS\$INSFMEM = ***** X 05
SS\$NORMAL = ***** X 05
SS\$NOSUCHNODE = ***** X 05
SS\$UNREACHABLE = ***** X 05
SS\$WRONGNAME = ***** X 05
SUPPRESS_AREA = 00000F38 RG 05
SYS\$CRELOG = ***** GX 05
SYS\$GETJPI = ***** GX 05
SYS\$PARSE = ***** GX 05
SYS\$WAITFR = ***** GX 05
SYSNODE_DESC = 00000020 R 03
TEMPRG = 00000J14
TEST_REACH = 00000F05 R 05
TR\$C_MAXHOR = 0000001C
TR\$C_NI_ALLEND1 = 040000AB
TR\$C_NI_ALLEND2 = 00000000
TR\$C_NI_ALLROU1 = 030000AB
TR\$C_NI_ALLROU2 = 00000000
TR\$C_NI_PREFIX = 000400AA
TR\$C_NI_PROT = 00000360
TR\$C_PRT_ECL = 0000001F
TR\$C_PRI_RTHRU = 0000001F
TR3\$\$\$_QUAL_MSG = 00000000

NETCNFACT
Symbol table

N 14
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 98
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

TR3\$\$\$ QUAL RTFLG = 00000000
TR3\$C_HSZ_DATA = 00000006
TR3\$C_MSG_DATA = 00000002
TR3\$C_MSG_HELLO = 00000005
TR3\$C_MSG_INIT = 00000001
TR3\$C_MSG_NOP2 = 00000008
TR3\$C_MSG_ROUT = 00000007
TR3\$C_MSG_STR2 = 00000058
TR3\$C_MSG_VERF = 00000003
TR3\$M_MSG_CTL = 00000001
TR3\$M_MSG_RTH = 00000002
TR3\$M_RTFLG_PH2 = 00000040
TR3\$M_RTFLG_RQR = 00000008
TR3\$M_RTFLG_RTS = 00000010
TR3\$R_QUAL = 00000000
TR3\$S_QUAL = 00000001
TR3\$S_QUAL_MSG = 00000001
TR3\$S_QUAL_RTFLG = 00000001
TR3\$S_RTFLG_012 = 00000003
TR3\$S_TR3MSG = 00000001
TR3\$V_MSG_CTL = 00000000
TR3\$V_MSG_RTH = 00000001
TR3\$V_RTFLG_012 = 00000000
TR3\$V_RTFLG_5 = 00000005
TR3\$V_RTFLG_7 = 00000007
TR3\$V_RTFLG_PH2 = 00000006
TR3\$V_RTFLG_RQR = 00000003
TR3\$V_RTFLG_RTS = 00000004
TR4\$\$\$ QUAL_ADDR = 00000000
TR4\$\$\$ QUAL_RTFLG = 00000000
TR4\$\$\$ QUAL_SCLASS = 00000000
TR4\$C_BCE_MID1 = 040000AB
TR4\$C_BCE_MID2 = 00000000
TR4\$C_BCR_MID1 = 030000AB
TR4\$C_BCR_MID2 = 00000000
TR4\$C_BCT3MULT = 00000008
TR4\$C_END_NODE = 00000003
TR4\$C_HIORD = 000400AA
TR4\$C_HSZ_DATA = 00000015
TR4\$C_MSG_BCEHEL = 00000000
TR4\$C_MSG_BCRHEL = 0000000B
TR4\$C_MSG_LDATA = 00000006
TR4\$C_MSG_RDATA = 00000002
TR4\$C_PRO_TYPE = 00000360
TR4\$C_RTR_LVL1 = 00000002
TR4\$C_RTR_LVL2 = 00000001
TR4\$C_T3MULT = 00000002
TR4\$C_VER_HIB = 00000000
TR4\$C_VER_LOWW = 00000002
TR4\$M_ADDR_AREA = 0000FC00
TR4\$M_ADDR_DEST = 000003FF
TR4\$M_RTFLG_INI = 00000020
TR4\$M_RTFLG_LNG = 00000004
TR4\$M_RTFLG_RQR = 00000008
TR4\$M_RTFLG_RTS = 00000010
TR4\$R_QUAL = 00000000
TR4\$S_ADDR_AREA = 00000006

TR4\$S_ADDR_DEST = 0000000A
TR4\$S_QUAL = 00000002
TR4\$S_QUAL_ADDR = 00000002
TR4\$S_QUAL_RTFLG = 00000001
TR4\$S_QUAL_SCLASS = 00000001
TR4\$S_RTFLG_01 = 00000002
TR4\$S_RTFLG_VER = 00000002
TR4\$S_SCLASS_57 = 00000003
TR4\$S_TR4MSG = 00000002
TR4\$V_ADDR_AREA = 0000000A
TR4\$V_ADDR_DEST = 00000000
TR4\$V_RTFLG_01 = 00000000
TR4\$V_RTFLG_INI = 00000005
TR4\$V_RTFLG_LNG = 00000002
TR4\$V_RTFLG_RQR = 00000003
TR4\$V_RTFLG_RTS = 00000004
TR4\$V_RTFLG_VER = 00000006
TR4\$V_SCLASS_1 = 00000001
TR4\$V_SCLASS_57 = 00000005
TR4\$V_SCLASS_BC = 00000004
TR4\$V_SCLASS_LS = 00000002
TR4\$V_SCLASS_METR = 00000000
TR4\$V_SCLASS_SUBA = 00000003
UCB\$C_LENGTH = 00000090
UCB\$Q_DWB_LIST = 00000090
UNAME = 00000044 R 02
UNAMES = 00000040 R 02
WQE\$B_EVL_DT1 = 0000001E
WQE\$B_EVL_DT2 = 0000001F
WQE\$B_EVL_PKT = 00000018
WQE\$B_EVL_CODE = 0000001C
WQE\$B_REQT = 00000012
XWB = 00000000
XWB\$B_ACCESS = 0000000B
XWB\$B_DATA = 0000005B
XWB\$B_FIPL = 0000001F
XWB\$B_LOGIN = 000000CC
XWB\$B_LPRNAM = 000000A4
XWB\$B_PRO = 0000005A
XWB\$B_RID = 0000006F
XWB\$B_RPRNAM = 000000B8
XWB\$B_SP3 = 0000006E
XWB\$B_STA = 0000001E
XWB\$B_TYPE = 0000000A
XWB\$B_X_FLW = 0000006C
XWB\$B_X_FLWCNT = 0000006D
XWB\$C_CONLNG = 000000A4
XWB\$C_CONLNG = 00000112
XWB\$C_DATA = 00000010
XWB\$C_LOGIN = 00000040
XWB\$C_LPRNAM = 00000014
XWB\$C_NDC_LNG = 00000020
XWB\$C_NUMSTA = 00000008
XWB\$C_RID = 00000010
XWB\$C_RPRNAM = 00000014
XWB\$C_STA_CAR = 00000002
XWB\$C_STA_CCS = 00000004

NETCNFACT
Symbol table

B 15
- Configuration data base access action

16-SEP-1984 01:13:22 VAX/VMS Macro V04-00
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1

Page 99
(48)

NE
VO

XWBS_C_STA_CIR = 00000003
XWBS_C_STA_CIS = 00000001
XWBS_C_STA_CLO = 00000000
XWBS_C_STA_DIR = 00000006
XWBS_C_STA_DIS = 00000007
XWBS_C_STA_RUN = 00000005
XWBSL_DEA_IRP = 00000104
XWBSL_FPC = 00000020
XWBSL_FR3 = 00000024
XWBSL_FR4 = 00000028
XWBSL_ICB = 0000010C
XWBSL_IRP_ACC = 00000080
XWBSL_LINK = 0000002C
XWBSL_ORGUCB = 00000010
XWBSL_PID = 00000034
XWBSL_VCB = 00000030
XWBSL_WLBL = 00000004
XWBSL_WLFL = 00000000
XWBSM_FLG_BREAK = 00000001
XWBSM_FLG_CLO = 00000200
XWBSM_FLG_I AVL = 00001000
XWBSM_FLG_SCD = 00000100
XWBSM_FLG_SDACK = 00000008
XWBSM_FLG_SDFL = 00004000
XWBSM_FLG_SDT = 00000080
XWBSM_FLG_SIACK = 00000004
XWBSM_FLG_SIFL = 00002000
XWBSM_FLG_SLI = 00000010
XWBSM_FLG_TBPR = 00000800
XWBSM_FLG_WBP = 00000040
XWBSM_FLG_WBUF = 00000002
XWBSM_FLG_WDAT = 00000400
XWBSM_FLG_WHGL = 00000020
XWBSM_PRO_CCA = 00000008
XWBSM_PRO_NAR = 00000010
XWBSM_PRO_NFC = 00000001
XWBSM_PRO_PH2 = 00000004
XWBSM_PRO_SFC = 00000002
XWBSM_STS_ASTPND = 00000400
XWBSM_STS_ASTREQ = 00000800
XWBSM_STS_CON = 00000010
XWBSM_STS_DIS = 00000008
XWBSM_STS_DTNAK = 00000100
XWBSM_STS_LINAK = 00000200
XWBSM_STS_NDC = 00001000
XWBSM_STS_OVF = 00000080
XWBSM_STS_RBP = 00000040
XWBSM_STS_SOL = 00000004
XWBSM_STS_TID = 00000001
XWBSM_STS_TLI = 00000002
XWBSM_STS_TMO = 00000020
XWBSQ_FORR = 00000014
XWBSQ_FREE_CXB = 00000118
XWBSR_CON_BLK = 000000A4
XWBSR_RUN_BLK = 000000A4
XWBS = 00000006
XWBS_COMING = 0000006E

XWBS_CON_BLK = 0000006E
XWBS_DATA = 00000010
XWBS_DT = 00000030
XWBS_FLG = 00000002
XWBS_FORK = 00000008
XWBS_FREE_CXB = 00000008
XWBS_LI = 00000030
XWBS_LOGIN = 0000003F
XWBS_LPRNAM = 00000013
XWBS_NDC = 00000020
XWBS_PRO = 00000001
XWBS_RID = 00000010
XWBS_RPRNAM = 00000013
XWBS_RUN_BLK = 00000064
XWBS_STS = 00000002
XWBS_XWB = 00000120
XWBS = 00000112
XWBS_DATA = 0000005C
XWBS_DT = 000000A4
XWBS_LI = 000000D4
XWBS_LOGIN = 000000CD
XWBS_LPRNAM = 000000A5
XWBS_RID = 00000070
XWBS_RPRNAM = 000000B9
XWBSV_FLG_BREAK = 00000000
XWBSV_FLG_CLO = 00000009
XWBSV_FLG_I AVL = 0000000C
XWBSV_FLG_SCD = 00000008
XWBSV_FLG_SDACK = 00000003
XWBSV_FLG_SDFL = 0000000E
XWBSV_FLG_SDT = 00000007
XWBSV_FLG_SIACK = 00000002
XWBSV_FLG_SIFL = 0000000D
XWBSV_FLG_SLI = 00000004
XWBSV_FLG_TBPR = 0000000B
XWBSV_FLG_WBP = 00000006
XWBSV_FLG_WBUF = 00000001
XWBSV_FLG_WDAT = 0000000A
XWBSV_FLG_WHGL = 00000005
XWBSV_PRO_CCA = 00000003
XWBSV_PRO_NAR = 00000004
XWBSV_PRO_NFC = 00000000
XWBSV_PRO_PH2 = 00000002
XWBSV_PRO_SFC = 00000001
XWBSV_STS_ASTPND = 0000000A
XWBSV_STS_ASTREQ = 0000000B
XWBSV_STS_CON = 00000004
XWBSV_STS_DIS = 00000003
XWBSV_STS_DTNAK = 00000008
XWBSV_STS_LINAK = 00000009
XWBSV_STS_NDC = 0000000C
XWBSV_STS_OVF = 00000007
XWBSV_STS_RBP = 00000006
XWBSV_STS_SOL = 00000002
XWBSV_STS_TID = 00000000
XWBSV_STS_TLI = 00000001
XWBSV_STS_TMO = 00000005

NETCNFACT
Symbol table

C 15
- Configuration data base access action 16-SEP-1984 01:13:22 VAX/VMS Macro V04-00 Page 100
5-SEP-1984 02:18:01 [NETACP.SRC]NETCNFACT.MAR;1 (48)

NE
VO

XWBSW_CI_PATH = 00000110
XWBSW_DECAY = 0000004E
XWBSW_DLY_FACT = 00000056
XWBSW_DLY_WGHT = 00000058
XWBSW_ELAPSE = 0000004A
XWBSW_FLG = 0000001C
XWBSW_LOCLNK = 0000003E
XWBSW_LOCSIZ = 00000040
XWBSW_PATH = 00000038
XWBSW_PROGRESS = 00000052
XWBSW_REFCNT = 0000000C
XWBSW_REMLNK = 0000003C
XWBSW_REMNOD = 0000003A
XWBSW_REMSIZ = 00000042
XWBSW_RETRAN = 00000054
XWBSW_R_REASON = 00000044
XWBSW_SIZE = 00000008
XWBSW_STG = 0000000E
XWBSW_TIMER = 00000050
XWBSW_TIM_ID = 00000048
XWBSW_TIM_INACT = 0000004C
XWBSW_X_REASON = 00000046
XWBSZ_NDC = 00000084

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
NET_IMPURE	00000130 (304.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
NET_PURE	000000A0 (160.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC LONG
\$RMSNAM	0000000F (15.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
NET_CODE	00001620 (5664.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
NET_LOCK_CODE	000000E9 (233.)	06 (6.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.08	00:00:00.33
Command processing	124	00:00:00.95	00:00:03.77
Pass 1	1711	00:00:43.98	00:01:02.76
Symbol table sort	1	00:00:05.75	00:00:06.68
Pass 2	1031	00:00:12.16	00:00:23.18
Symbol table output	32	00:00:00.72	00:00:01.42
Psect synopsis output	4	00:00:00.05	00:00:00.13
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2935	00:01:03.69	00:01:38.31

The working set limit was 900 pages.
243327 bytes (476 pages) of virtual memory were used to buffer the intermediate code.
There were 210 pages of symbol table space allocated to hold 3730 non-local and 302 local symbols.

4101 source lines were read in Pass 1, producing 54 object records in Pass 2.
86 pages of virtual memory were used to define 64 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SHRLIB]NMALIBRY.MLB;1	1
\$255\$DUA28:[SHRLIB]EVCDEF.MLB;1	1
\$255\$DUA28:[NETACP.OBJ]NETDRV.MLB;1	2
\$255\$DUA28:[NETACP.OBJ]NET.MLB;1	24
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	4
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	21
TOTALS (all libraries)	53

3953 GETS were required to define 53 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:NETCNFACT/OBJ=OBJ\$:NETCNFACT MSRCS:NETCNFACT/UPDATE=(ENHS:NETCNFACT)+EXECMLS/LIB+LIB\$:NET/LIB+LIB\$:NETDRV/LIB+SHRLIB\$

0274 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

